MacTech Magazine
Vol. 19, No. 10 • October 2003

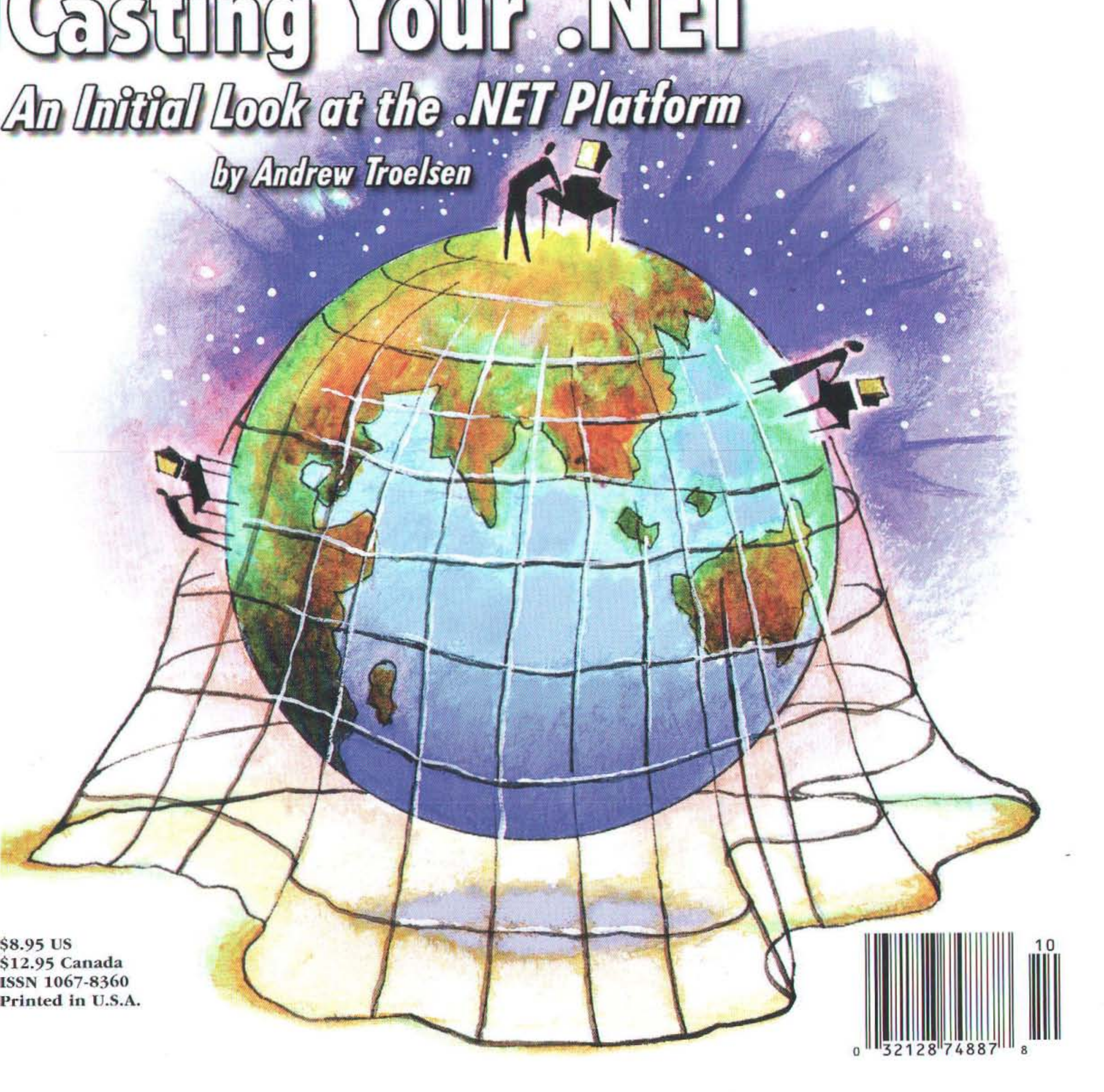
MacTech®

The Journal of Macintosh Technology and Development

Casting Your .NET

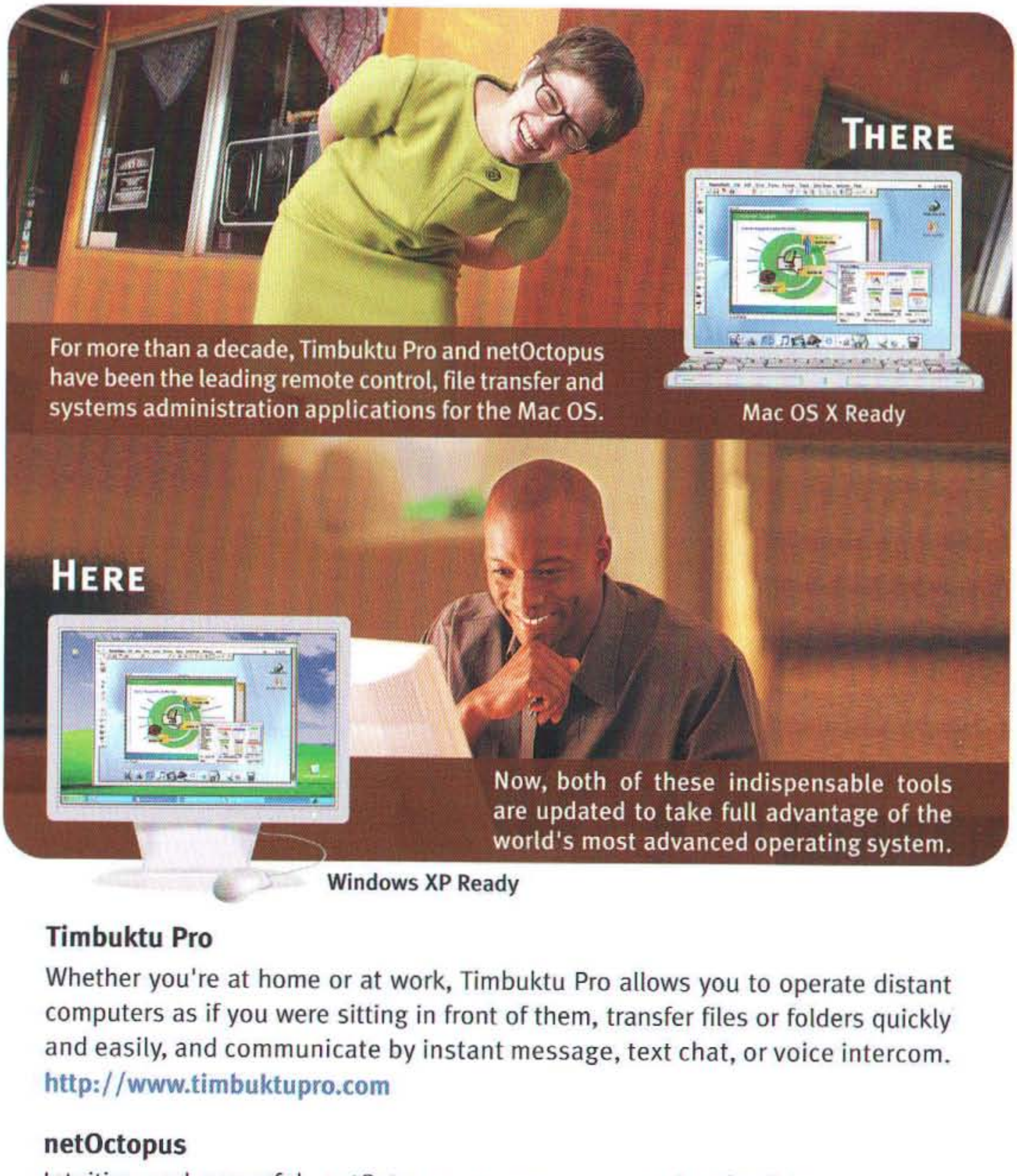
An Initial Look at the .NET Platform

by Andrew Troelsen



$8.95 US
$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.

Stay In Control Wherever You Go.



For more than a decade, Timbuktu Pro and netOctopus have been the leading remote control, file transfer and systems administration applications for the Mac OS.

Mac OS X Ready

Now, both of these indispensable tools are updated to take full advantage of the world's most advanced operating system.

Windows XP Ready

Timbuktu Pro

Whether you're at home or at work, Timbuktu Pro allows you to operate distant computers as if you were sitting in front of them, transfer files or folders quickly and easily, and communicate by instant message, text chat, or voice intercom.
http://www.timbuktupro.com

netOctopus

Intuitive and powerful, netOctopus can manage a network of ten or 10,000 computers. Inventory computers, software and devices on your network; distribute software; configure remote computers; and create custom reports on the fly.
http://www.netoctopus.com

Learn more, try it, or buy it online. Call us at 1-800-485-5741.

timbuktu® · netOctopus®

netopia.

CUSTOMIZE

YOUR DATABASE SERVER WITH THE

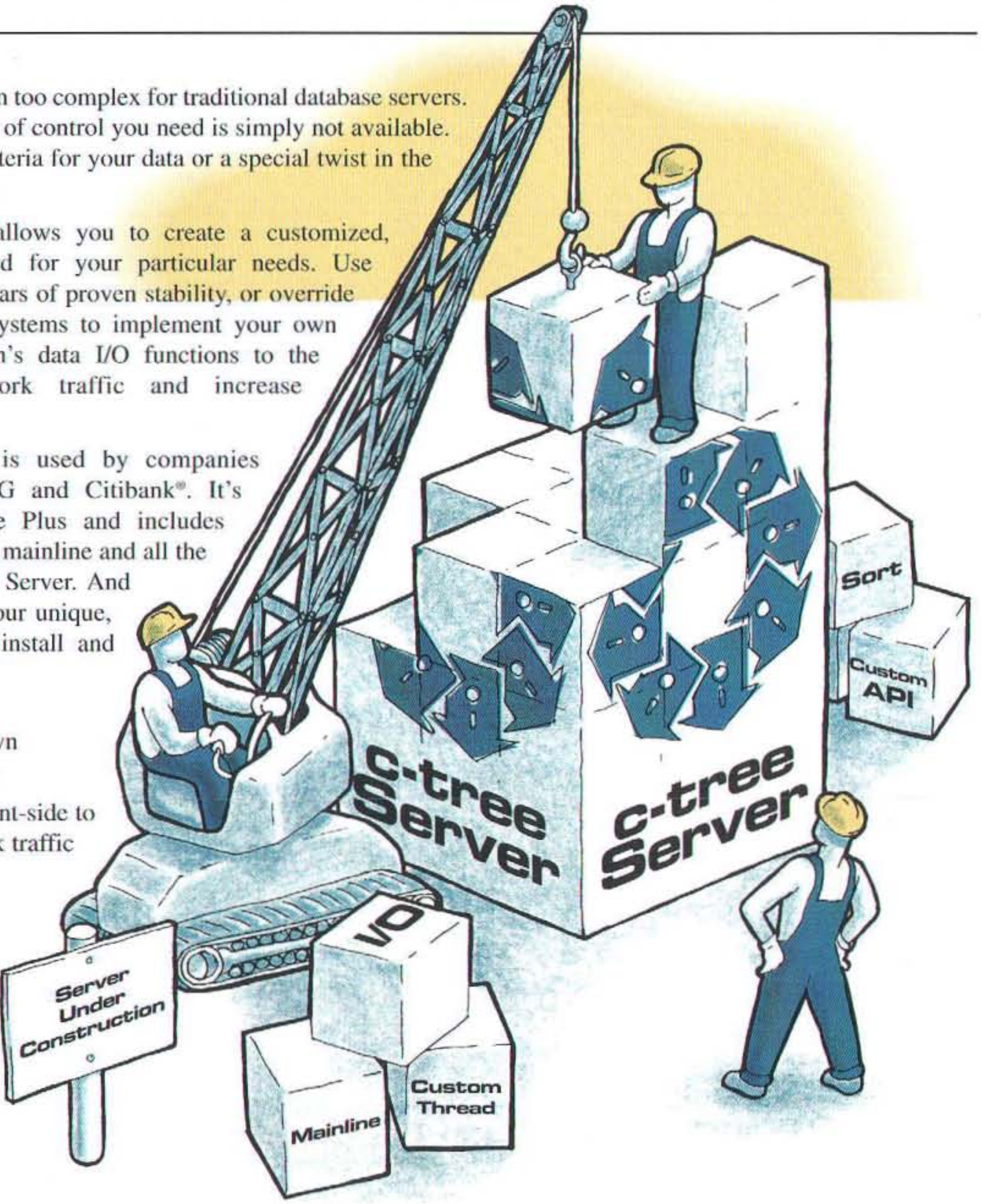
C-TREE® SERVER SDK

Today's database demands are often too complex for traditional database servers. The functionality and precise level of control you need is simply not available. Perhaps you need alternate sort criteria for your data or a special twist in the threading or communication logic.

FairCom's c-tree® Server SDK allows you to create a customized, industrial-strength server designed for your particular needs. Use FairCom's kernel, with over 20 years of proven stability, or override functionality within specific subsystems to implement your own subtleties. Move your application's data I/O functions to the server-side to decrease network traffic and increase performance!

FairCom's c-tree Server SDK is used by companies worldwide such as Software AG and Citibank®. It's integrated seamlessly into c-tree Plus and includes complete source code to the server mainline and all the interface subsystems to the c-tree Server. And best of all, once you've created your unique, customized server, it is easy to install and administer: no DBA required!

- Enhance our server with your own custom server-side functionality
- Move functionality from the client-side to the server-side to reduce network traffic and increase performance
- Modify or replace entire server subsystems
- Complete source for the server mainline, key server subsystems, and client-side
- Flexible OEM licensing



Visit www.faircom.com/ep/mt/sdk today to take control of your server!

FairCom®
www.faircom.com

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

DBMS Since 1979 • 800.234.8180 • info@faircom.com

HELPING FILEMAKER PRO CUSTOMERS SCALE TO NEW HEIGHTS.

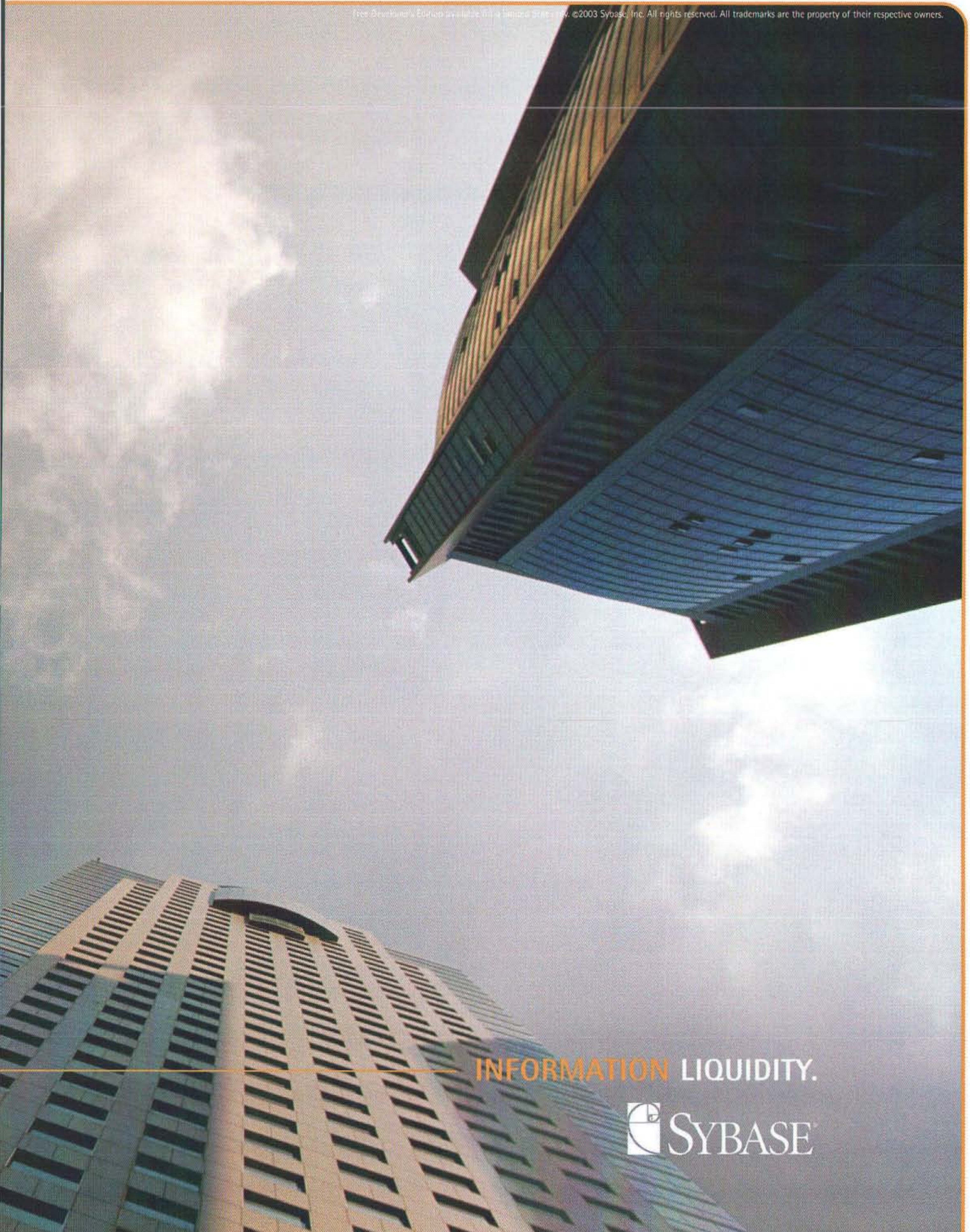
Want to enhance the performance of FileMaker Pro on MAC OS X? The same database engine that runs Wall Street can help you do just that.

ASE 12.5 increases the reliability, availability and scalability of your FileMaker Pro application. It provides better data integrity, world-class security and the ability to handle thousands of transactions per minute. It'll also give your users the power of SQL queries.

ASE 12.5 for MAC OS X is yet one more example of how Sybase is helping today's leading companies achieve Information Liquidity: a highly profitable state where all of your information is transformed into real economic value.

A FREE Developer's Edition download is available online at sybase.com/filemaker.

SYBASE INTEGRATION TECHNOLOGIES. EVERYTHING WOR


INFORMATION LIQUIDITY.
SYBASE
BETTER WHEN EVERYTHING WORKS TOGETHER.™

MacTech®

The Journal of Macintosh Technology & Development

A publication of **XPLAIN**CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS	E-Mail/URL
Orders, Circulation, & Customer Service	cust_service@mactech.com
Press Releases	press_releases@mactech.com
Ad Sales	ad_sales@mactech.com
Editorial	editorial@mactech.com
Programmer's Challenge	prog_challenge@mactech.com
Online Support	online@mactech.com
Accounting	accounting@mactech.com
Marketing	marketing@mactech.com
General	info@mactech.com
Web Site (articles, info, URLs and more...)	http://www.mactech.com

The MacTech Editorial Staff

Publisher • Neil Ticktin

Editor-in-Chief • Dave Mark

Managing Editor • Jessica Stubblefield

Regular Columnists

Getting Started
by Dave Mark

QuickTime ToolKit
by Tim Monroe

Mac OS X Programming Secrets
by Scott Knaster

Reviews/KoolTools
by Michael R. Harvey

Patch Panel
by John C. Welch

Section 7
by Rich Morin

Untangling the Web
by Kevin Hemenway

John and Pals' Puzzle Page
by John Vink

Regular Contributors

Vicki Brown, Erick Tejkowski, Paul E. Sevinç

MacTech's Board of Advisors

Jordan Dea-Mattson, Jim Straus and Jon Wiederspan

MacTech's Contributing Editors

- Michael Brian Bentley
- Vicki Brown
- Marshall Clow
- John. C. Daub
- Tom Djajadiningrat
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Sun
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Rich Morin
- John O'Fallon, Maxum Development
- Will Porter
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Chuck Von Rospach, Plaidworks

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • Darryl Smart

Marketing Manager • Nick DeMello

Account Executive • Lorin Rivers
adsales@mactech.com • 800-5-MACDEV

Events Manager • Susan M. Worley

International • Rose Kemps

Network Administrator • David Breffitt

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane, Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park, and Alan Carsrud

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

Contents

October 2003 • Volume 19, Issue 10

Help Authoring

Getting Started

Software Marketing

QuickTime Toolkit

Cover Story

Book Review

Untangling the Web

Reviews

Mac OS X Programming Secrets

Cocoa Tip

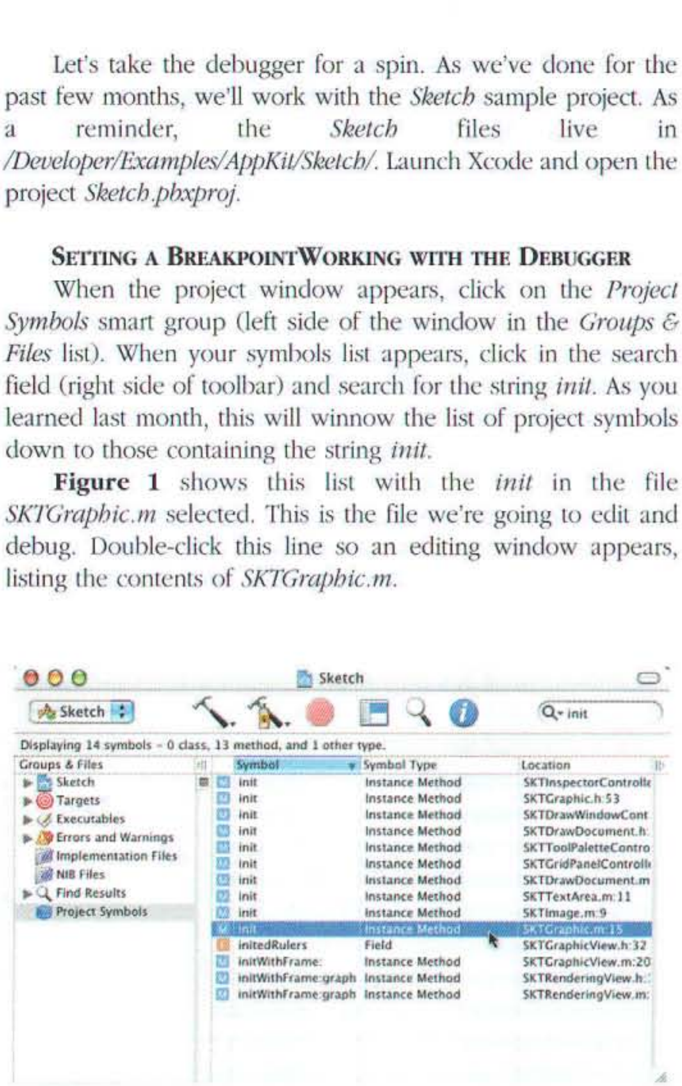
Getting Started: More on Xcode

Let's take the debugger for a spin. As we've done for the past few months, we'll work with the *Sketch* sample project. As a reminder, the *Sketch* files live in */Developer/Examples/AppKit/Sketch/*. Launch Xcode and open the project *Sketch.pbxproj*.

Setting a BreakpointWorking with the Debugger

When the project window appears, click on the *Project Symbols* smart group (left side of the window in the *Groups & Files* list). When your symbols list appears, click in the search field (right side of toolbar) and search for the string *init*. As you learned last month, this will winnow the list of project symbols down to those containing the string *init*.

Figure 1 shows this list with the *init* in the file *SKTGraphic.m* selected. This is the file we're going to edit and debug. Double-click this line so an editing window appears, listing the contents of *SKTGraphic.m*.



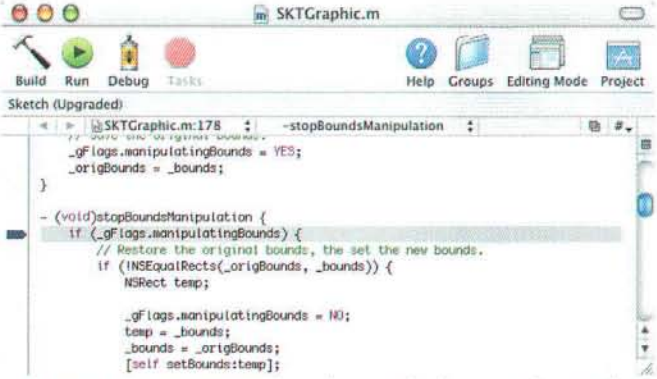
Displaying 14 symbols - 0 class, 13 method, and 1 other type.

Symbol	Symbol Type	Location
init	Instance Method	SKTInspectorControlle
init	Instance Method	SKTGraphic.h:53
init	Instance Method	SKTDrawWindowCont
init	Instance Method	SKTDrawDocument.h:
init	Instance Method	SKTToolPaletteContro
init	Instance Method	SKTGridPanelControll
init	Instance Method	SKTDrawDocument.m
init	Instance Method	SKTTextArea.m:11
init	Instance Method	SKTImage.m:9
init	Instance Method	SKTGraphic.m:15
initedRulers	Field	SKTGraphicView.h:32
initWithFrame:	Instance Method	SKTGraphicView.m:20
initWithFrame:graph	Instance Method	SKTRenderingView.h:
initWithFrame:graph	Instance Method	SKTRenderingView.m:

***Figure 1.** Search the Project Symbols for files containing init.*

Setting a Breakpoint

Scroll about one quarter of the way down the source file and click on line 178. As a reminder, the navigation bar is the strip immediately above the main editing pane. The line number follows the file name, which is immediately to the right of the navigation arrows (see **Figure 2**). Once you locate line 178, set a breakpoint by clicking in the left column, to the left of the line in the editing window. Take a look at **Figure 2**. Note that the breakpoint arrow appears to the left of line 178. This means that the debugger, once started, will stop execution just *before* it executes the if statement on line 178.



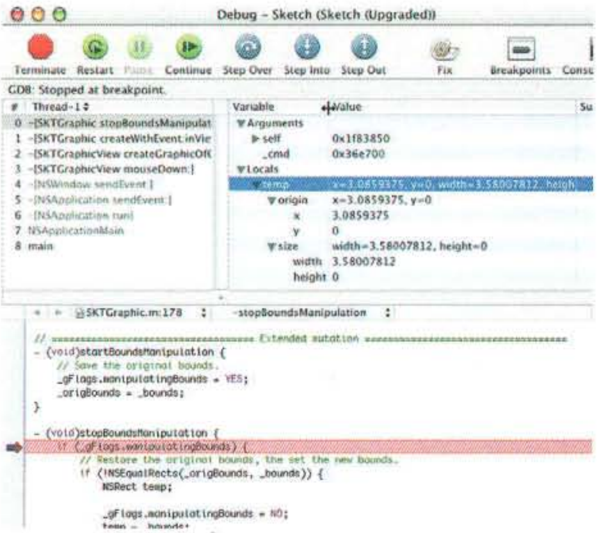
```
    _gFlags.manipulatingBounds = YES;
    _origBounds = _bounds;
}

- (void)stopBoundsManipulation {
    if (_gFlags.manipulatingBounds) {
        // Restore the original bounds, the set the new bounds.
        if (!NSEqualRects(_origBounds, _bounds)) {
            NSRect temp;

            _gFlags.manipulatingBounds = NO;
            temp = _bounds;
            _bounds = _origBounds;
            [self setBounds:temp];
```

***Figure 2.** Line 178 is selected. Note the line number in the navigation bar and the breakpoint set on the left.*

To start the debugger, click the *Debug* icon in the *SKTGraphic.m* editing window's toolbar. Depending on things like the speed of your machine, how much of the program is already compiled, etc., this may take a bit. Be patient.

Once your code is built, the *Sketch* app is launched. Click on the rectangle tool and drag out a rectangle. As you release the mouse button, the debugger will hit that breakpoint and bring the *Debug* window to the front. **Figure 3** shows the *Debug* window when you hit the breakpoint.

Dave Mark is a long-time Mac developer and MacTech contributor. Author of more than a dozen books on various Mac-development topics, Dave is all about Xcode these days. Last month's column focused on the editor interface as well as fix-and-continue. This month's installment will take the debugger through a few of its paces. and explore Xcode's code completion feature.



```
GDB: Stopped at breakpoint.

#  Thread-1                               Variable      Value                                              Su
0  -[SKTGraphic stopBoundsManipulat       ▼Arguments
1  -[SKTGraphic createWithEvent:inVie       ▶self       0x1f83850
2  -[SKTGraphicView createGraphicOfC         _cmd       0x36e700
3  -[SKTGraphicView mouseDown:]           ▼Locals
4  -[NSWindow sendEvent:]                   ▼temp       x=3.0859375, y=0, width=3.58007812, heigh
5  -[NSApplication sendEvent:]               ▼origin    x=3.0859375, y=0
6  -[NSApplication run]                           x     3.0859375
7  NSApplicationMain                              y     0
8  main                                      ▼size      width=3.58007812, height=0
                                              width     3.58007812
                                              height    0
```

```
SKTGraphic.m:178    -stopBoundsManipulation

// ==================================== Extended mutation ====================================
- (void)startBoundsManipulation {
    // Save the original bounds.
    _gFlags.manipulatingBounds = YES;
    _origBounds = _bounds;
}

- (void)stopBoundsManipulation {
    if (_gFlags.manipulatingBounds) {
        // Restore the original bounds, the set the new bounds.
        if (!NSEqualRects(_origBounds, _bounds)) {
            NSRect temp;

            _gFlags.manipulatingBounds = NO;
            temp = _bounds;
```

***Figure 3.** TheDebug window, showing the program paused at a breakpoint. Note the display of Locals in the upper-right pane.*

Take a look at the upper-right pane in **Figure 3.** Notice the display of variables. You can manipulate the column widths by dragging on the splits between the *Variable*, *Value*, and *Summary* column headers. I maximized the width of the *Value* column in **Figure 3**. You can click on the disclosure triangles to reveal fields within structs. For example, *origin* has an *x* and a *y* component. The *origin* line lists the values of all the *origin* fields. Open the triangle and each field gets its own line.

If you double-click on the *x* value, the value of *x* turns into a text editing field allowing you to modify the value of *x*. Just as you'd expect.

But if you double-click on the *origin* value (the one that lists *x* and *y* values separated by a comma), the *origin* formatter will appear in a text editing field. In this case, the formatting string is:

```
x=%x%, y=%y%%origin%, %size%
```

Double-click the *size* value column and the *size* formatter will appear:

```
width=%width%, height=%height%
```

Let's change the formatter. Change it to read:

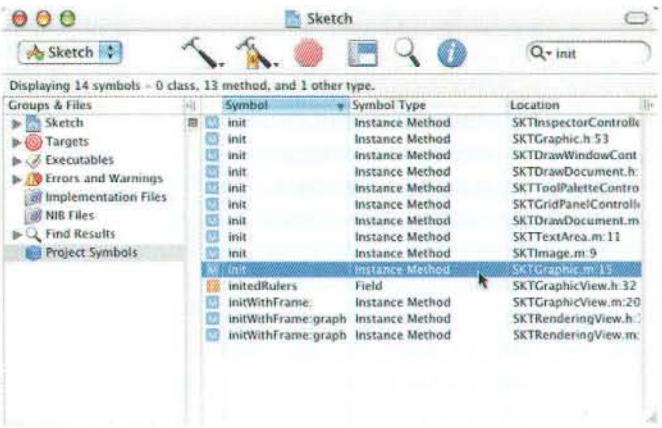
```
w=%width%, h=%height%
```

When you finish the edit (hit enter), the change you made will propagate up through both the display of *size* and any other displays that reference *size*. To see this for yourself, click the Arguments triangle, then the self triangle, then the _bounds triangle to reveal *origin* and *size* within *_bounds*. On my display (see **Figure 4**), the *_bounds* value field is:

End the compromise
with perfect quality and file size.
Produce highly compressed, lossless QuickTime videos for delivering screen capture training videos, animated text and graphics.
TechSmith
ENSHARPEN™
Video Codec
Try it FREE today at www.techsmith.com

```
x=50, y=63, w=107, h=169
```

When we changed one *size* formatter, the change propagated to all fields that displayed a value of type *size*. Data formatters are tied to type definitions, which are global.



Displaying 14 symbols – 0 class, 13 method, and 1 other type.

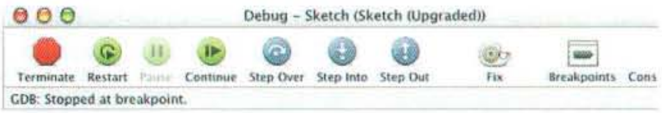
Symbol	Symbol Type	Location
init	Instance Method	SKTInspectorControlle
init	Instance Method	SKTGraphic.h:53
init	Instance Method	SKTDrawWindowCont
init	Instance Method	SKTDrawDocument.h:
init	Instance Method	SKTToolPaletteContro
init	Instance Method	SKTGridPanelControlle
init	Instance Method	SKTDrawDocument.m
init	Instance Method	SKTTextArea.m:11
init	Instance Method	SKTImage.m:9
init	Instance Method	SKTGraphic.m:15
initedRulers	Field	SKTGraphicView.h:32
initWithFrame:	Instance Method	SKTGraphicView.m:20
initWithFrame:graph	Instance Method	SKTRenderingView.h:
initWithFrame:graph	Instance Method	SKTRenderingView.m:

Figure 4. *In this shot, notice that self._bounds.size uses the modified size formatter.*

If you are unfamiliar with formatters, grab your favorite C text and do a bit of digging. Just like a formatter in a *printf()*, Xcode formatters allow you to specify how the debugger displays your variables. There's even an API so you can write your own formatters. See the Xcode release notes for more info.

Debug Controls

Figure 5 shows the debugger controls you'll find at the top of your *Debug* window. Chances are, if you've ever used a debugger before, you'll recognize most of them. *Terminate* terminates the process being debugged, *Restart* terminates the process and restarts it from the beginning. *Pause* pauses and *Continue* continues execution from the current stopping point. *Step Over* executes the next line of code without stepping into any functions, while *Step Into* steps to the next line of code, even if it means stepping into a function. *Step Out* completes the current function and stops immediately after the function's return in the calling function.

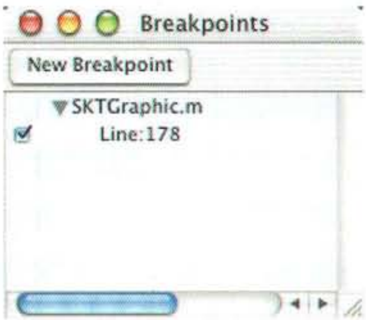


GDB: Stopped at breakpoint.

Figure 5. *The debugger controls at the top of the Debug window.*

The (in my opinion) coolest icon of the bunch, the *Fix* scotch tape dispenser is the equivalent of selecting *Fix* from the *Debug* menu. As I mentioned in last month's column, there appears to be a glitch with this icon in the Jaguar build of Xcode which I believe is fixed in the Panther build. Once Panther is officially released, I'll definitely be working exclusively with the Panther tools, so we'll get a sense of what's what in the current release of the tools.

The *Breakpoints* icon brings up the *Breakpoints* window (see **Figure 6**), which lists all your current breakpoints, organized by source file. You can turn breakpoints on and off using the checkbox associated with each breakpoint.



▼SKTGraphic.m
☑ Line:178

Figure 6. *The Breakpoints window.*

The *Console Drawer* icon opens a console drawer below the *Debug* window, allowing you to directly enter *gdb* commands. This is extremely cool, especially if you come from the *Unix* world and learned debugging using gdb, adb, or the equivalent.

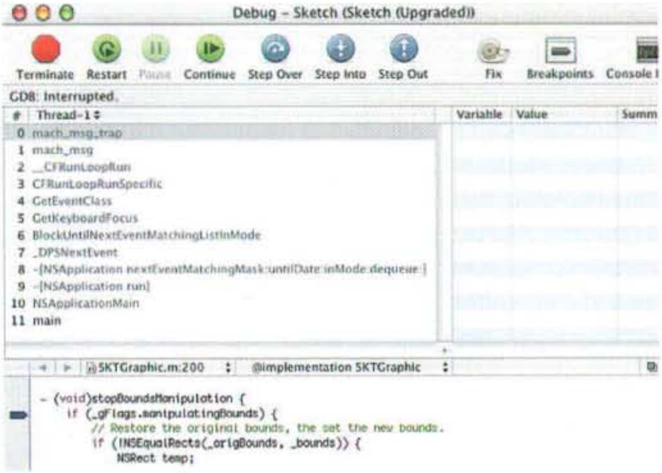
You've got the gdb doc on your hard drive:

/Developer/Documentation/DeveloperTools/gdb/gdb/gdb_toc.html

Once More Into the Breach

Let's take one more spin through the debugger. If *Sketch* is currently debugging, click the *Terminate* icon. Now restart *Sketch* by clicking the *Debug* icon. When *Sketch* starts running, do *not* do anything in *Sketch*, just use the dock to return to Xcode. Back in Xcode, select *Debug* - ... from the *Window* menu to bring the *Debug* window to the front. Now, click on the *Pause* button to stop *Sketch* at whatever line was executing when you clicked the *Pause* button.

Figure 7 shows the event trace pane which is a stack showing the call sequence with main at the bottom and the current function at the top (in this case, the trap *mach_msg_trap*). Basically, this is what your app looks like when it is in an idle state, waiting for an event to happen.



GDB: Interrupted.

#	Thread-1
0	mach_msg_trap
1	mach_msg
2	__CFRunLoopRun
3	CFRunLoopRunSpecific
4	GetEventClass
5	GetKeyboardFocus
6	BlockUntilNextEventMatchingListInMode
7	_DPSNextEvent
8	-[NSApplication nextEventMatchingMask:untilDate:inMode:dequeue:]
9	-[NSApplication run]
10	NSApplicationMain
11	main

SKTGraphic.m:200 @implementation SKTGraphic

```
- (void)stopBoundsManipulation {
    if (_gFlags.manipulatingBounds) {
        // Restore the original bounds, the set the new bounds.
        if (!NSEqualRects(_origBounds, _bounds)) {
            NSRect temp;
```

***Figure** 7. The event trace pane shows call sequence stack.*

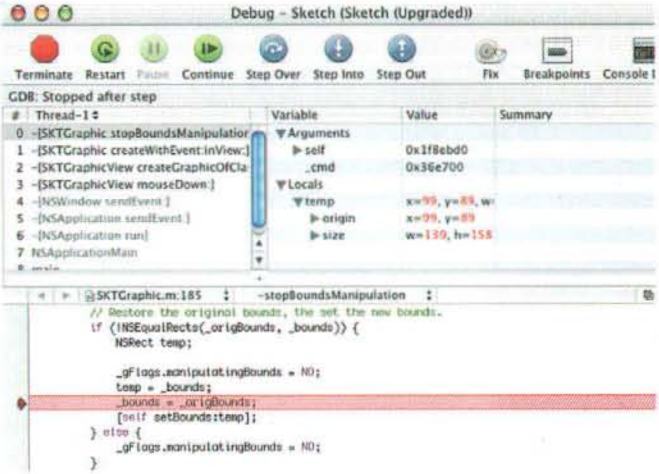
Next, click the *Continue* icon to get *Sketch* back up and running. Back in *Sketch*, click on the rectangle tool and drag out a rectangle. When you let go, you should pop back into the debugger and your familiar breakpoint.

Click *Step Into* a few times until this line is highlighted:

```
temp = _bounds;
```

Remember, since this line is highlighted it has not yet executed. Click *Step Into* one more time. As you can see in **Figure 8**, when you execute this line and assign a value to *temp*, the *temp* fields displayed in the Value column are displayed in red, showing that they have changed.

If you click *Step Into* one more time, you'll change *bounds*. *To see this change, you'll need to open the self triangle. self._bounds* will turn red and *temp* will return to black.



GDB: Stopped after step

#	Thread-1
0	-[SKTGraphic stopBoundsManipulatior
1	-[SKTGraphic createWithEvent:inView:]
2	-[SKTGraphicView createGraphicOfCla
3	-[SKTGraphicView mouseDown:]
4	-[NSWindow sendEvent:]
5	-[NSApplication sendEvent:]
6	-[NSApplication run]
7	NSApplicationMain
8	main

Variable	Value	Summary
▼Arguments		
▶self	0x1f8ebd0	
_cmd	0x36e700	
▼Locals		
▼temp	x=99, y=89, w	
▶origin	x=99, y=89	
▶size	w=139, h=158	

SKTGraphic.m:185 -stopBoundsManipulation

```
        // Restore the original bounds, the set the new bounds.
        if (!NSEqualRects(_origBounds, _bounds)) {
            NSRect temp;

            _gFlags.manipulatingBounds = NO;
            temp = _bounds;
            _bounds = _origBounds;
            [self setBounds:temp];
        } else {
            _gFlags.manipulatingBounds = NO;
        }
```

***Figure 8.** When we change the value in temp, the temp fields turn red.*

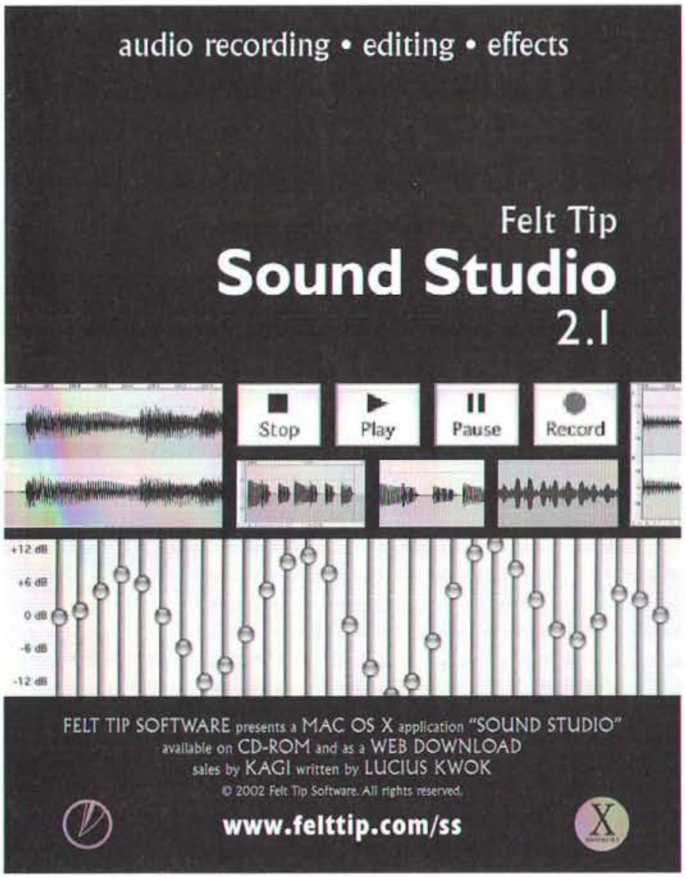
For our final trick, click on the *Console Drawer* icon. Scroll to the bottom of the window till you see this prompt:

```
(gdb)
```

At the prompt, type the command *continue* and hit return. This is exactly as if you had clicked the *Continue* icon in the toolbar. You are now back in *Sketch* and you can drag out some more shapes, as you like.

Till Next Month...

Whelp, I'm out of space again. <sigh>. There's so much more I want to talk about. Next month, we're going to take the debugger through its paces and we'll also play around with a feature called *code completion*, one of my very favorite parts of XcodeYou know, there are way more things I want to write about than there is space in the magazine. Note to self – see about adding more pages to mag. Ah, well. I've got a cool column idea for next month. Not sure I can pull it off. We'll See see. Look for you then…☺



MAC OS X PROGRAMMING SECRETS

By Scott Knaster

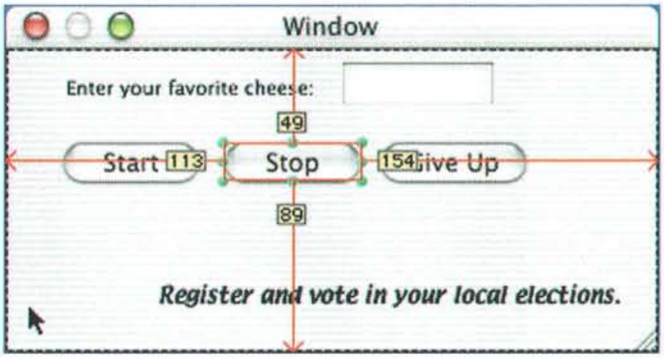
Interface Builder Tips and Tricks

When you're programming with Cocoa, you spend plenty of time in Interface Builder, instantiating objects, positioning things just right in windows, and making connections. In this month's column, we're going to shine the light of day on a few valuable tips and shortcuts for making yourself more productive when using Interface Builder.

Designing for Success

Most of the intricate work in Interface Builder is in the design window, which is where you add objects from palettes, lay them out, resize them, and so on. In this section, we'll discuss some tips for use in design windows.

The first trick is useful for figuring out precise distances between objects while looking at them in the design window. If you select an object, then hold down the Option key, you'll see the exact distances from the borders of the object to the edges of the window (see **Figure 1**).

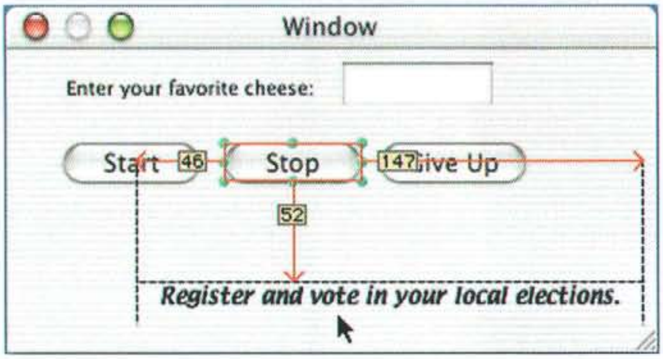


Enter your favorite cheese:

Register and vote in your local elections.

Figure 1. *Hold down Option to see distances.*

But wait, that's not all. By moving the mouse pointer while holding down Option, you can have Interface Builder show you the distance between objects, rather than just the distance to the window edges. In **Figure 2**, the pointer is on the static text object, and the window shows the distances from the selection to each edge of that object.



Enter your favorite cheese:

Register and vote in your local elections.

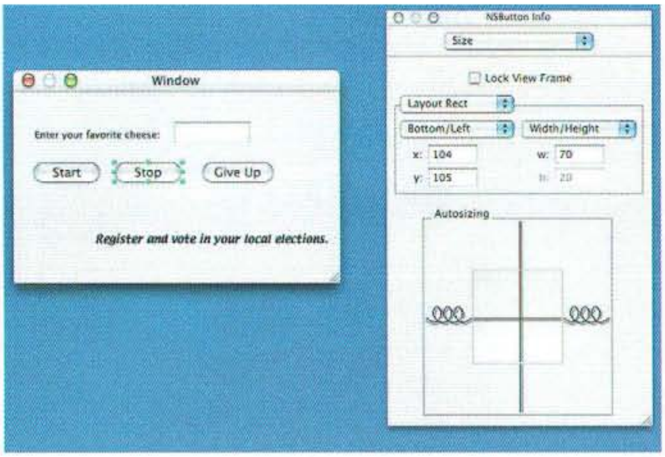
Figure 2. *Hold down Option to see distances.*

You can use another Interface Builder feature to make objects automatically resize or reposition themselves when you change the size of their enclosing window. For example, let's say you want to preserve the proportional distance between the three buttons in the window above: if you widen the window, you want the distance between the buttons to increase. To make this happen, you can create springs between the objects. To do this, select each object in turn, open the size inspector, and click that line that you want to become a spring (a proportional relationship that changes automatically when you resize the window). **Figure 3** shows what this looks like for the "Stop" button in the window above: we've clicked on its connection lines to make them into springs. We'll do the same for all 3 buttons.

Now, when we resize the window, if we hold down the Control key, the springs will cause the distances between objects to change proportionally. If we want the springs to be ignored, we can just resize without holding down Control.

Scott Knaster has been writing about Macs for as long as there have been Macs. Scott's books ***How To Write Macintosh Software*** and ***Macintosh Programming Secrets*** were required reading for Mac programmers for more than a decade. Scott wrote developer books for General Magic and worked on Mac software for Microsoft. Scott's books have been translated into Japanese and Pascal. Scott has every issue of Mad magazine, which explains a lot.

UNFAIR



x: 104
y: 105
w: 70
h: 20

Figure 3. *Stop button has springs on its left and right.*

Speaking of the size inspector, it's even smarter than you might have realized. You can type math expressions into the coordinate fields and let Interface Builder evaluate them for you. For example, if you want to add 32 pixels to an object's width, you can just click in the w field, select after the current value, then type "+32" and press return. Neato! Apple even provides code to help you implement this trick in your own fields. Take a look at /Developer/Examples/InterfaceBuilder/BusyPalette for details.

Another fun trick you can use with design windows is to see all the rectangles that Interface Builder uses to determine distances between objects. By using the Layout --> Show Layout Rectangles menu item, or just pressing command-L, you can see all the layout rectangles at once, as shown in **Figure 4**.

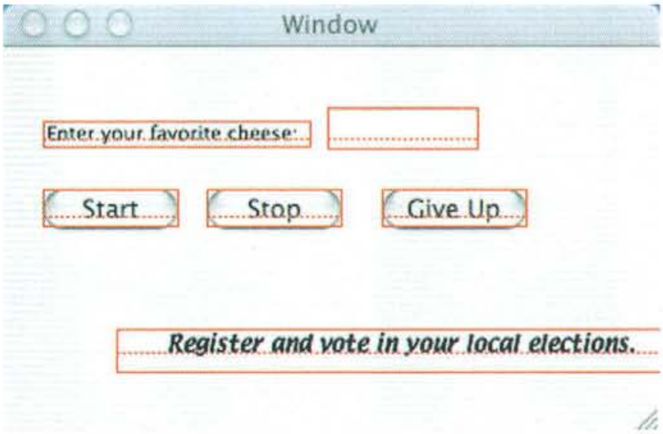

Window
Enter your favorite cheese:
Start
Stop
Give Up
Register and vote in your local elections.

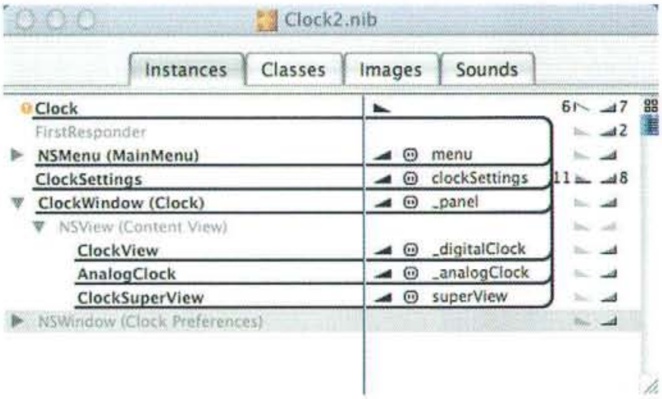
Figure 4. Design window with all layout rectangles shown.

Interface Builder tips aren't all about layout. For example, there's a quick way to set the tab order for objects in the window. You'll want to do this to allow users to tab from one text field to the next. The tab order starts with the initial first responder, and continues from there by following each object's nextKeyView field.

To start setting the order, select the object you want to be the initial first responder and choose Layout --> Keyboard Navigation --> Make Initial First Responder. Then, to set the tab order, choose Layout --> Keyboard Navigation --> Show Keyboard Check, Control-drag from one object to the next, then click Connect in the connection inspector. That will set the tab connection between the objects. Note that when you have keyboard check turned on, the word *tab* appears at the bottom left of the window. That reminds you that you can press tab to test the tab order.

Cool Stuff in Other Windows, Too

Not all the neat tricks are in the design window – here are a couple you can use in the document window. If you click the tiny control on the right side of the window, the object instances are displayed in an outline view, with the number of connections, both into and out of the object. If you click on one of these numbers, the connections appear in the window, as you can see in **Figure 5.**



```
Clock                                           6     7
FirstResponder                                        2
▶ NSMenu (MainMenu)              menu
ClockSettings                    clockSettings   11   8
▼ ClockWindow (Clock)            _panel
  ▼ NSView (Content View)
      ClockView                  _digitalClock
      AnalogClock                _analogClock
      ClockSuperView             superView
▶ NSWindow (Clock Preferences)
```

Figure 5. *Outline view shows connections to an object.*

You can use this view to delete connections between objects. If you hold down Control and point at a line that connects two objects, the pointer changes to a tiny pair of scissors, and clicking (you guessed it) cuts the connection.

If you just want to learn more about the connected objects, click on the name of the connected field, such as *menu* or *_panel* in Figure 5. The connected objects will appear on the screen with a line between them.

There's More (Always)

You can find more fun and timesaving tips like these by reading the informative FAQ that comes with Interface Builder. Get to the FAQ by choosing FAQ in Interface Builder's Help menu. There's even a sense of humor in the FAQ: in the section on modifier keys, right after the descriptions of what happens if you press Command, Option, and Control, the FAQ tells you that you can use the Left Pedal to disengage the clutch. Cool.

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."
– MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."
– MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"
– Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."
– Bill Goodman, author of Smaller Installer and Compact Pro

"Resorcerer Rocks! Buy it, you will NOT regret it."
*– Joe Zobkiw, author of **A Fragment of Your Imagination***

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

EDITORS' CHOICE AWARD MACUSER 1993 WINNER

Accelerated for Power Macintosh

Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater, 1.5MB RAM, CD-ROM

Standard price: $256 (decimal)
Website price: $128 - $256
(Educational, quantity, or other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMÆSTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New in 2.0:

- **Very fast, HFS browser for viewing file tree of all volumes**
- **Extensibility for new Resorcerer Apprentices (CFM plug-ins)**
- **New AppleScript Dictionary ('aete') Apprentice Editor**
- **MacOS 8 Appearance Manager-savvy Control Editor**
- **PowerPlant text traits and menu command support**
- **Complete AIFF sound file disassembly template**
- **Big-, little-, and even mixed-endian template parsing**
- **Auto-backup during file saves; folder attribute editing**
- **Ships with PowerPC native, fat, and 68K versions**

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

By Andrew Troelsen, Minneapolis, Minnesota

An Initial Look at the .NET Platform

Exploring .NET development on Mac OS X

WELCOME, SKEPTICAL READERS...

Building raw application infrastructure by hand is a time consuming, error prone and painful process. Who among us really wants to create yet another linked list, database access framework or user interface library? Thankfully, numerous platform development kits exist today. By leveraging APIs (*application programming interfaces*) such as J2SE/J2EE, Cocoa, and Carbon we are able to keep focused on the programming task at hand by leveraging a set of canned functionality.

As a software professional the chances are good that you are aware of a new development platform created by Microsoft named .NET (pronounced, *dot net*). As well, you may have also heard of a plethora of new programming languages that target this new framework such as C# (*see sharp*), Visual Basic.NET, managed C++ and Pascal.NET (to name a few). However, given .NET's point of origin (Microsoft), you may have naturally assumed that .NET development is solely the activity of Windows programmers. If you did make this assumption, you might be quite surprised to learn that C# and the .NET platform are alive and well under Mac OS X, as well as other Unix-based systems.

In this new MacTech series, it is my goal to provide a guided tour of the .NET platform as seen through the eyes of a Macintosh developer. Over the articles to come, you will learn how to configure your development machine(s) with the necessary infrastructure required to build various types of .NET applications using multiple .NET distributions and .NET-aware programming languages.

To begin our journey however, we need to gain a high-level perspective of .NET itself, which is the point of this first article. Please be *very aware* that all of the topics examined here will be further explained and expanded upon in subsequent articles. Given this friendly disclaimer, kick back with your beverage of choice and let's get to know the .NET platform.

THE PHILOSOPHY OF THE .NET PLATFORM

The .NET platform is a software development API that was officially released to the programming world circa 2001. Despite the name (which tends to conjure up visions of Web-enabled front ends), the .NET platform can be used to build traditional desktop applications, Terminal applications *as well as* server-side Web applications and XML Web services.

Like other frameworks, .NET provides a class library that allows developers to build various forms of applications. Not only does this base class library encapsulate various primitives such as threads, object serialization services, file IO, and interaction with various hardware devices (such as printers and handheld devices), but it also provides support for a number of services required by most real world applications (e.g. GUI toolkits, XML manipulation, network protocols and so forth).

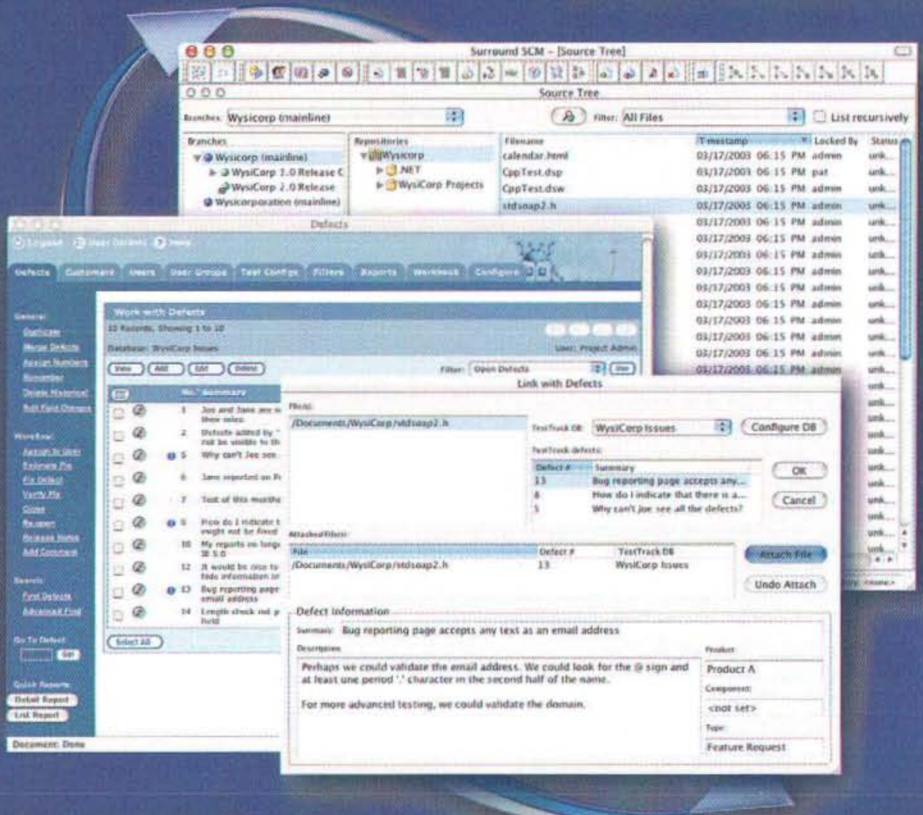
However, unlike many other frameworks, the .NET base class libraries may be accessed using numerous programming languages. Not only can developers pick their programming language of choice when building .NET solutions, but they may also create a single .NET application using multiple languages. For example, you could make use of C# to build a desktop application which communicates with a XML Web services written using Visual Basic.NET. Even more interestingly, .NET supports cross-language inheritance, cross-language exception handling and cross-language interface implementation. Given these traits, you will do well to view .NET as a *language-agnostic* framework.

In addition to being a language-agnostic development platform, .NET is also *platform-agnostic*. This tidbit is the one aspect of .NET that tends to surprise most developers. While it is true that Microsoft Corporation has never been in the business of building software that can (easily) run of multiple operating systems, the tides have turned with the release of the .NET platform. Like J2SE, the .NET platform allows you to build software that can execute on multiple operating systems without modification. Therefore, if you build a .NET-based program on your Macintosh G5, you are able distribute and execute this application on any operating system hosting the .NET runtime.

Andrew Troelsen is a seasoned .NET developer who has authored numerous books on the topic, including the award winning *C# and the .NET Platform*. He is employed as a full-time .NET trainer and consultant for Intertech Learning (www.intertechlearning.com) and spends his idle moments at work showing off his fancy new Macintosh PowerBook G4 to his envious cohorts. You can contact Andrew at atroelsen@mac.com.

Complete Source Control and Defect Management for Mac OS X

Seapine Software™
changing the world of software development



Surround SCM – [Source Tree]

Source Tree

Branches: Wysicorp (mainline) Filter: All Files ☐ List recursively

Branches	Repositories	Filename	Timestamp	Locked By	Status
Wysicorp (mainline)	Wysicorp	calendar.html	03/17/2003 06:15 PM	admin	unk...
WysiCorp 1.0 Release C	.NET	CppTest.dsp	03/17/2003 06:15 PM	pat	unk...
WysiCorp 2.0 Release	WysiCorp Projects	CppTest.dsw	03/17/2003 06:15 PM	admin	unk...
Wysicorporation (mainline)		stdsoap2.h	03/17/2003 06:15 PM	admin	unk...

Defects

Link with Defects

Files: /Documents/WysiCorp/stdsoap2.h

TestTrack DB: WysiCorp Issues (Configure DB)

TestTrack defects:

Defect #	Summary
13	Bug reporting page accepts any...
8	How do I indicate that there is a...
5	Why can't joe see all the defects?

Attached Files:

File	Defect #	TestTrack DB
/Documents/WysiCorp/stdsoap2.h	13	WysiCorp Issues

Defect Information

Summary: Bug reporting page accepts any text as an email address

Description: Perhaps we could validate the email address. We could look for the @ sign and at least one period '.' character in the second half of the name.

For more advanced testing, we could validate the domain.

Product: Product A

Component: <not set>

Type: Feature Request

Effective source code control and defect tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- Fast and secure remote access to your source files and defects — work from anywhere
- Advanced branching simplifies managing multiple versions of your products
- Link code changes with defects and change requests — know who changed what, when, and why
- Scalable and reliable cross-platform, client/server solutions support Mac OS X, Windows, Linux, and Solaris
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget

Seapine Software Product Lifecycle Management
Award winning, easy-to-use software development tools

Seapine Surround SCM

Seapine TestTrack PRO

software development 13th annual product excellence award JOLT

Download Surround SCM and TestTrack Pro at www.seapine.com or call 1-888-683-6456

all product names listed herein are registered trademarks of their respective owners. All rights reserved.

The Building Blocks of .NET

Clearly, a language and platform neutral platform such as .NET is founded upon a number of well-defined rules and underlying technologies. Given this, our next task is to preview (the operative word being *preview*) some key and interrelated specifications that make it all possible: the Common Language Infrastructure, the Common Type System and the Common Language Specification.

The Common Language Infrastructure (CLI)

First up, we have a specification termed the Common Language Infrastructure, or simply CLI. This particular aspect of .NET is the foundation upon which the platform and language independent nature of .NET is established. The CLI is a fairly large specification that formalizes several critical elements of the platform, including:

- An abstract runtime engine termed the Virtual Execution System (VES), which is capable of hosting .NET compliant code libraries.
- The internal structure of the .NET compliant code libraries (which go by the name *assemblies*) to be hosted by the VES.
- A unified type system termed the Common Type System (CTS), which describes the rules used to compose the programming types contained within .NET assemblies.
- The syntax and semantics of the low-level language used to implement the types contained within .NET assemblies, termed the Common Intermediate Language (CIL).

The VES is a hypothetical execution engine that sits on top of the actual underlying operating system (such as Mac OS X or Windows XP). The primary role of the VES is to locate, load, execute and manage .NET code libraries (a.k.a. assemblies) on your behalf. As well, the VES takes care of a number of related details such as automatic memory management (via garbage collection), language integration, ensuring type safety and verifying the correctness of the assembly's CIL code (more on CIL a bit later in this article).

One a related note, if you do have some experience using Microsoft's official .NET platform, you may have heard of another term: the Common Language Runtime (CLR). Basically, the CLR is the formal name given to a VES implementation that is specific to the Windows family of operating systems. Given that we are not all that interested in the Windows-centric implementation of the CLI's execution engine, I'll stick to the more generic term 'VES' when referring to the execution engine of .NET.

The Common Type System (CTS)

The next building block of the .NET platform, the Common Type System, or CTS, is concerned with formalizing the structure of the types contained within the code libraries themselves. Simply put, the CTS specifies all of the possible data types (e.g. Booleans, strings, integers, etc) and programming constructs (e.g., iteration constructs, virtual methods, overloaded operators, type constructors and so on) supported by the .NET runtime.

One of the most fundamental rules of the CTS is the fact that every .NET type derives from a common base class named System.Object. For your edification, **Listing 1** shows the C# definition of this top-most class type:

Listing 1:The C# definition of the ultimate .NET base class

```
namespace System
{
  public class Object
  {
     public Object();
    public virtual Boolean Equals(Object obj);
    public virtual Int32 GetHashCode();
    public Type GetType();
     public virtual String ToString();
     protected virtual void Finalize();
     protected Object MemberwiseClone();
  }
}
```

At this point in the game, don't worry if you cannot make heads or tails of previous code listing. The point here is that the CTS is the aspect of the CLI that describes a set of rules that constitute a well-defined .NET type, one such rule being the fact that System.Object is the cosmic super class. As you would hope, other rules of the CTS and the role of System.Object (not to mention the C# programming language) will be examined in great detail at a later time.

The Common Language Specification (CLS)

Understand that a particular .NET language (such as COBOL.NET) is not required to support each and every aspect formalized by the CTS. For example, although the CTS does define how unsigned data types are to be represented in CIL and processed by the runtime engine, not all .NET languages support unsigned data. Given that .NET is a language-neutral platform, it would be quite helpful to have a well-defined baseline of functionality. The Common Language Specification (CLS) is effectively a subset of the CTS, which defines a set of data types and programming constructs that all .NET languages can agree upon.

When you build .NET assemblies that only makes use of the guidelines expressed by the CLS, you can rest assured that your code library can be correctly used by any .NET language (such code is termed 'CLS compliant'). On the other hand, if you build a .NET code library that makes use of features outside the realm of the CLS (but within the confines of the CTS), you have just created an assembly that may only be completely usable from select .NET programming languages.

To solidify the concepts presented thus far, figure 1 illustrates how various .NET programming languages ultimately target the VES and CTS/CLS specifications.

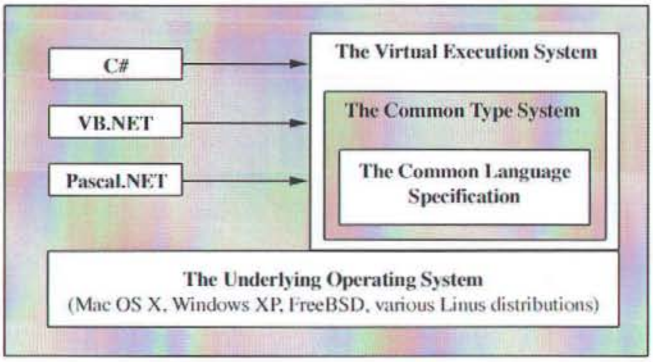

C#
VB.NET
Pascal.NET
The Virtual Execution System
The Common Type System
The Common Language Specification
The Underlying Operating System
(Mac OS X, Windows XP, FreeBSD, various Linus distributions)

Figure 1. *.NET programming languages target the .NET platform, which is built upon numerous specifications.*

Now that you have a better idea regarding the role of the CLI, let's dig into some further details of the .NET type system.

The Type System of .NET

Every programming language (and therefore every programming platform) exposes a 'type system'. This term boils down to the set of syntactically well-formed user defined types (UDTs) supported by said language (or platform). To illustrate a well-know type system, consider Java. As of Java 1.4, the type system consists of class types and interface types. Beginning with Java 1.5, the type system will be expanded to include C-style enumerations. In a similar vein, the C++ programming language supports a type system consisting of the set {class, enumeration, structure}, with interfaces being simulated via class types and a corresponding typedef or two.

.NET also defines a set of possible types, which have you have seen is officially termed the CTS. As you would expect from a modern day API, this type system is extremely well organized and completely object oriented (including support for a relatively new programming paradigm termed *aspect oriented programming*).

To illustrate the symmetry of the .NET type system, ponder the fact that every single UDTs provided by the .NET platform, including any custom software entities you create yourself, will fall into one of the following categories:

- Class
- Interface
- Structure
- Enumeration
- Delegate

While these type categories will be fully explained in later articles, **Table 1** offers high-level definitions for each member of the .NET type system.

.NET Type	Meaning in Life
Class	Class types are heap allocated entities that may be extended and modified by derived classes. .NET classes are very similar to Java, C++ and Objective C class types. As you know, classes form the basis of the famed 'pillars of OOP' (encapsulation, inheritance and polymorphism).
Interface	An interface is a collection of abstract members grouped by a particular name. Using interfaces, developers are able to establish polymorphism between types *not* related by classical inheritance.
Structure	Structures are lightweight, stack-allocated entities that are well suited for modeling numeric, geometric and atomic user defined types.
Enumeration	Enumerations (or simply, *enums*) are a named collection of name/value pairs.
Delegate	By far and away, .NET delegates are the most confounding 'type of type' to contend with. For now I will simply define .NET delegates as nothing more than a type-safe and object oriented function pointer (more details to come in later articles).

Table 1. *The .NET type system*

Given that .NET provides many thousands of predefined types (collectively termed the 'base class libraries' or simply

BCL), you will be happy to know that like-minded types are organized into a conceptual boundary termed a *namespace*.

Understanding .NET Namespaces

The namespace concept is not new. The Java, C++ and Smalltalk languages have used similar (but not identical) concepts to partition types for years. In a nutshell, a .NET namespace is a collection of semantically related types. For example, the System.Collections namespace contains a set of classes, interfaces, enumerations, structures and delegates that represent 'types that manage other types' (e.g. linked lists, array lists, FILO / FIFO types and so on). The System.Data namespace defines numerous types that have to do with database manipulations. **Table 2** documents some (but by no means all) interesting .NET namespaces.

.NET namespace	Meaning in Life
System	The System namespace defines various core types you will use on a day-to-day basis. Here you will find the native .NET data types, core exceptions and numerous utility types.
System.Collections	As mentioned, this namespace defines types that can contain and manage other types (e.g. ArrayList, Stack, Queue and so on).
System.Data	The .NET platform defines numerous data-centric namespaces that allow you to interact with various relational databases. System.Data is the lowest common denominator for data access under .NET.
System.Drawing	The drawing-centric namespaces allow you to render graphical data to a surface (such as window of a desktop application or a region of memory).
System.IO	Defines types to work with various input/output streams.
System.Reflection	The reflection namespaces allow you to programmatically discover the functionality of types (as well as generate new types) at runtime.
System.Runtime.Remoting	.NET defines numerous namespaces which allow you to build distributed applications.
System.Web	The Web-centric namespaces allow you to build Web-based applications (ASP.NET) and XML Web services.
System.Windows.Forms	Windows Forms provides types that allow you to build traditional desktop applications.
System.Xml	As you would hope, the .NET platform defines numerous namespaces that facilitate the manipulation of XML data.

Table 2. *A sampling of .NET namespaces.*

Interacting with the Base Class Libraries

Regardless of which sort of application you intend to build, you will inevitably interact with numerous .NET namespaces. However, the way you programmatically denote the namespace you wish to consume will differ based upon your .NET programming language of choice. If you are using C#, you would make use of the using keyword to specify the namespaces you wish to use. For example consider **Listing 2:**

Listing 2: Specifying namespaces in C#

```
// My way cool C# program.
using System;
using System.Windows.Forms;
using System.Drawing;

public class MyApp
{
  //ToDo:Add some interesting code...
}
```

On the other hand, if you would rather build your application using Visual Basic.NET, you use the Imports keyword to achieve the same affect, as seen in **Listing 3:**

Listing 3: Specifying namespaces in VB.NET

```
' My way cool Visual Basic.NET program.
Imports System
Imports System.Windows.Forms
Imports System.Drawing

Public Class MyApp
  'ToDo:Add some interesting code...
End Class
```

In either case, once you declare the namespaces you wish to consume, you are now able to access each of the types defined within the System, System.Windows.Forms and System.Drawing namespaces.

The Fully Qualified Name of a Type

It is worth pointing out that explicitly specifying the namespaces you wish to interact with using a language specific keyword is technically optional. To understand why this is the case, allow me to define the term *fully qualified name*. As you have already seen, all .NET types live within a namespace definition. Thus, if you wish to make use of the ArrayList type defined in the System.Collections namespace, you might author the following C# code (**Listing 4**):

Listing 4: Using the ArrayList type via namespace references

```
using System.Collections;

public class MyClass
{
  public void SomeMethod()
  {
     // Add some data to an ArrayList.
     ArrayList arList = new ArrayList();
     arList.Add(12);
     arList.Add("Don't forget wife's birthday");
  }
}
```

W WINDOWS | **M** MACINTOSH | **X** MAC OS X

Thinkfree

ThinkfreeOffice

COMPATIBLE WITH MICROSOFT WORD, EXCEL AND POWERPOINT

WORDPROCESSOR • SPREADSHEET • PRESENTATION GRAPHICS

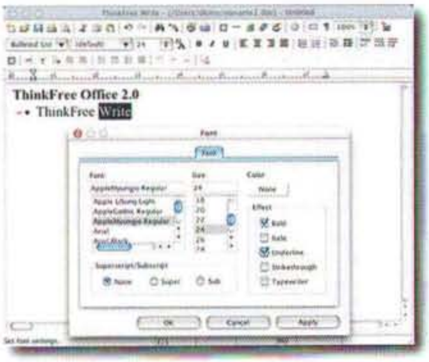
The Affordable Office Alternative!

The Affordable Office Alternative!
ThinkFree Office Suite Includes:
thinkfree
Office
Macworld
Compatible with Microsoft Word, Excel and PowerPoint

Three High-Performance Applications

Thinkfree Write

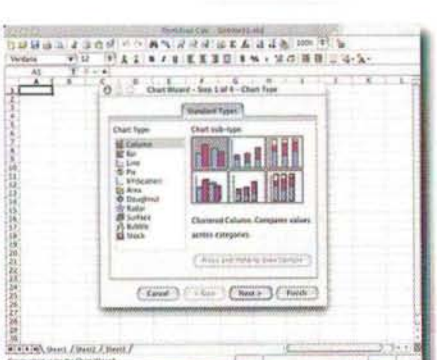
Thinkfree Write is a powerful word processing application that enables you to create rich, professional quality documents and Web pages. You can insert tables, images, and clipart, or even apply custom layouts to your document...then effortlessly proofread your work with the easy-to-use spelling and auto-correction features.


ThinkFree Office 2.0

- ThinkFree Write

Thinkfree Calc

Thinkfree Calc is a full-featured, easy-to-use spreadsheet application that can easily tackle the most complex analytical tasks with over 40 charts and 300 function capabilities. Thinkfree Calc opens, edits, and saves directly into the Microsoft Excel (.xls) format, so users can seamlessly share documents and collaborate with Microsoft Office users.



Thinkfree Show

Thinkfree Show enables you to create high-impact presentations including animation effects, drawings, images, clipart, and other graphic features. Thinkfree Show opens, edits and saves directly into the Microsoft PowerPoint (.ppt) format.

CyberdrivePlus

A free, one-year subscription to CyberdrivePlus is also included. CyberdrivePlus provides you with secure, Internet file storage and free online software upgrades!

ONLY $49^95

Macworld

"Thinkfree is a best-of-breed program that will exceed your expectations."
— *Jeffery Battersby*

MacAddict RATED GREAT

"Thinkfree Office is the next best thing and then some."
— *Deborah Shadovitz*

macHOME

"Thinkfree Office is an impressive attempt to crack the seemingly impenetrable productivity market."
— *Chris Ward*

amazon.com

The Macintosh Superstore! MAC Solutions 1-800-873-3726

Apple Store

MacMall

Apple Specialist

J&R

COMPUSA.com

Small Dog Electronics

Fry's ELECTRONICS

ZONES

Thinkfree and Thinkfree Office are trademarks of Thinkfree Corporation. All other trademarks and/or registered trademarks are properties of their respective owners.

While you will typically always prefer to explicitly list the namespaces used by a given source code file, the previous code is nothing more than a short hand notation for the following code in **Listing 5** (note the lack of a C# using statement):

Listing 5: Using ArrayList via the fully qualified name

```
public class MyClass
{
  public void SomeMethod()
  {
    // Add some data to an ArrayList.
    System.Collections.ArrayList arList
      = new System.Collections.ArrayList();
    arList.Add(12);
    arList.Add("Don't forget wife's birthday");
  }
}
```

As you can gather, the fully qualified name of a type is simply a term used to identify a type prefixed by its defining namespace. While making use of fully qualified names may make your code painfully well documented, hand cramps are sure to follow. Given this, most programmers prefer to list the set of namespaces used by a given file using a language specific keyword.

In terms of performance, there is absolutely no difference whatsoever between identifying a type with the fully qualified name or via a namespace reference. As you will see in just a bit, the .NET platform *always* refers to a type using its fully qualified name. In this light, the namespace concept is simply a typing time saver.

If you have a background in the J2SE platform, understand that the C# using and VB.NET Imports keyword do not support a Java-style * notation. If you wish to access nested namespaces, you must list out each nested namespace in turn. Thus, the following C# code is syntactically correct (**Listing 6**):

Listing 6: Nested namespaces must be listed explicitly

```
// Making use of three drawing related namespaces.
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
```

While the following Java-like notation (**Listing 7**) will cause a compiler error issued by the C# compiler:

Listing 7: Java-like * syntax not supported in C#!

```
//Oops! Remember, C# is not Java!
using System.Drawing.*;
```

Creating Custom .NET Namespaces

In addition to programming with the namespaces defined by the .NET base class libraries, you may define your own custom namespaces as well. By way of a simple example, assume you have defined two class types and one enumeration. To group these items into a namespace definition, you could write the following C# code (**Listing 8**):

Listing 8: Namespaces contain any number of .NET types

```
namespace MyCustomTypes
{
  public class MyClass {…}
  public class MyOtherClass {…}
  public enum MyEnum {…}
}
```

The corresponding VB.NET code example is quite similar (**Listing 9**):

Listing 9: Ditto.

```
Namespace MyCustomType
  Public Class MyClass
  …
  End Class
  Public Class MyOtherClass
  …
  End Class
  Public Enum MyEnum
  …
  End Enum
End Namespace
```

If you wish to use these types from another namespace, you would simply make use of the correct language specific keyword used to reference an external namespace. For example, in C# we would write the following (**Listing 10**):

Listing 10: Referencing a custom namespace in C#

```
using MyCustomTypes;

namespace MyNewNamespace
{
  public class MyNewClass
  {
    // You can now make use of any of the types in
    // the MyCustomTypes namespace
  }
}
```

So much for our initial high-level overview of the .NET type system. At this point in the game simply understand that the .NET type system consists of the set {class, interface, enumeration, structure, delegate} and that semantically related types are grouped together using the namespace concept. Next up, let's check out further details regarding the language agnostic nature of the .NET platform.

.NET is a Language Agnostic Platform

As mentioned, one distinguishing aspect of the .NET platform is the fact that developers may make use of any combination of programming languages to access the types of the base class libraries. Microsoft itself has created five .NET-aware programming languages: C#, J#, VB.NET, managed C++ and JScript.NET. While C# and VB.NET tend to steal most of the spotlight, there are literally dozens of languages which can be used to interact with the .NET libraries. Consider the partial list offered by **Table 3** (please note that the exact URLs are subject to change in the future).

.NET Aware Programming Language	Meaning in Life
Component Pascal http://www.citi.qut.edu.au/research/plas/projects/index.jsp .	Pascal language bindings for.NET
Eiffel http://archive.eiffel.com/doc/manuals/technology/dotnet/eiffelsharp	Eiffel language bindings for .NET.
Fortran http://www.lahey.com/lf70/lf70.htm	Fortran language bindings for .NET.
Perl and Python http://aspn.activestate.com/ASPN/NET	Perl and Python to .NET language bindings are also available.
Scheme http://rover.cs.nwu.edu/~scheme	The Scheme language is supported under the .NET platform as well.
SmallScript (a.k.a., S#) http://www.smallscript.net	Yes, even Smalltalk supports binding to the .NET platform.

***Table 3.** A partial listing of .NET aware programming languages*

When developer's first learn about the language agnostic aspect of the .NET platform, various questions tend to surface. One such question is more practical in nature, and tends to go something like this: "If all languages targeting .NET have access to the same libraries, why do we need more than one language?" The corresponding practical answer is based on the simple fact that programmers tend to be very particular which their syntactic preferences (myself included).

Some developers love to see curly brackets and semi-colons galore (as seen in C-based languages such as Objective C, C++ and Java). Others prefer more 'human-readable' syntactic tokens as found in the BASIC family of languages. Given the philosophy of .NET, we area able to stay true to our syntactic preferences while accessing the .NET base class libraries.

Furthermore, remember the obvious fact that each programming language has its own set of strengths and weaknesses. Some languages have excellent native support for advanced mathematical processing, while others excel in the realms of financial or logical processing. Again, using .NET you can capitalize upon the unique aspects of a given language while interacting with the fabric provided by the .NET base class libraries.

The Role of .NET Assemblies

Obviously, knowing how to develop with a particular .NET programming language (or two) is only part of the process. Once you have authored your .NET source code files that define your namespaces and their contained types, they are fed into the related complier to produce a binary file termed an assembly. Assemblies are platform-neutral binaries blobs that contain generic descriptions of types and their implementation that are hosted by the VES. To be more specific, a .NET assembly is a unit of deployment that is composed of the following three ingredients:

- Common Intermediate Language (CIL)
- Type metadata
- Assembly-level metadata (termed the manifest)

Again, later articles will drill into the details of .NET assemblies, CIL code and metadata. However to prime the pump, let's see a simple example which illustrates the basic composition of a .NET assembly. Assume we have the following C# source code file seen in **Listing 11** (note that by convention, C# code files end in a *.cs file extension):

Listing 11:A trivial .NET source code file (in C#)

```
// MyCSharpApp.cs
using System;

namespace AllMyStuff
{
  public class MyCSharpApp
  {
    public void EchoToTerminal(string message)
    {
      Console.WriteLine("You said: {0}",
        message);
    }
  }
}
```

Even if you have never seen C# code before this article, the previous code example should not raise too many eyebrows. Here we see a class type with a single method named

EchoToTerminal(), which takes a string type as its only parameter. This method is implemented to take the incoming argument and print the contents to the Macintosh Terminal. Notice that the WriteLine() method of the System.Console type takes a literal string argument containing an optional (and strange) token {0}, which is dynamically replaced by the specified second argument. For the time being, simply regard this notation as a kinder, simpler version of the C printf() function.

Once this source code file has been processed by the C# compiler, we end up with an assembly, which for the sake of argument, we will name MyCSharpApp.dll. Although the file extension of a .NET assembly can technically be anything you choose, you are likely to find two common extensions in use today. If you have a .NET assembly ending in the extension *.exe, you are looking at an executable program that can be directly loaded by the VES. On the other hand, assemblies which end with a *.dll file extension are dynamic link libraries which cannot run in and of themselves. Rather, *.dll files are loaded on demand my other running applications (very similar to the *.dylib modules found under Mac OS X).

The Guts of a .NET Assembly

As you will see in the next article, .NET distributions ship with a tool named ildasm (the CIL disassembler) which can be used to view the underlying CIL code, type metadata and assembly manifest contained within a given .NET assembly. For our current discussion, we are only concerned with the internal representation of the EchoToTerminal() member, as seen in **Listing 12:**

Listing 12: EchoToTerminal() expressed in CIL code

```
.method public hidebysig
 instance void EchoToTerminal
 (class System.String message) cil managed
{
  .maxstack  8
  ldstr "You said: {0}"
  call void [mscorlib]
   System.Console::WriteLine(class System.String)
  ret
}
```

Without getting too bogged down into the details at this point, simply notice how the C# implementation of EchoToTerminal() has been processed by the C# compiler into terms of CIL code. As you may have already guessed, the ldstr CIL operational code loads a string for use, while call is used to invoke a type member (in this case, the WriteLine() method of System.Console) with the currently loaded arguments. Finally, the ret operational code is the CIL-savvy way to return from a method.

Now assume we have implemented the same exact class type using VB.NET as shown in **Listing 13** (by convention, VB.NET source code files end with a *.vb file extension):

Listing 13: A trivial .NET source code file (in VB.NET)

```
'MyVbNetApp.vb
Imports System

Namespace AllMyStuff
  Public Class MyVbNetClass
    Public Sub EchoToTerminal(message as String)
      Console.WriteLine("You said: {0}", message)
    End Sub
  End Class
End Namespace
```

Once this source code file is processed by the VB.NET compiler, we can again use ildasm to view the resulting CIL generated by the VB.NET compiler (**Listing 14**):

Listing 14: EchoToTerminal() expressed in CIL code (yet again)

```
.method public hidebysig
 instance void EchoToTerminal
 (class System.String message) cil managed
{
  .maxstack  8
  ldstr "You said: {0}"
  ldarg.1
  call void
  [mscorlib]System.Console::WriteLine(
  class System.String, class System.Object)
  ret
}
```

As you have just seen, regardless of the fact that different .NET-aware programming languages define unique keywords and programming idioms, the associated compiler transforms these tokens into a common language (CIL) which is injected into the binary assembly. While it is true that individual .NET code compilers are free to take some liberties regarding the transformation of their syntactic tokens into CIL, the fact remains that as far as the .NET runtime is concerned, CIL is the only language worth speaking and the only true language processed by the .NET runtime engine.

To wrap things up for this first article, we close with a further examination regarding the platform-agnostic nature of the .NET platform.

.NET is a Platform Agnostic Platform

Having a language-agnostic .NET assembly does little good to anyone unless it can be loaded, executed and processed by a runtime engine. As noted earlier in this article, the Virtual Execution System (VES) is the entity in charge of hosting a valid .NET assembly. In addition to being a language-agnostic platform, .NET is also *platform-agonistic*. If you have any experience with anything related to Microsoft, this last statement is sure to resonate as a flat out lie. However, as strange as it may seem, while Microsoft was developing their official .NET distribution, a team of engineers created an alternative implementation of the CLI (termed the Shared Source CLI) that may be compiled and run on numerous operating systems (including of course, Mac OS X).

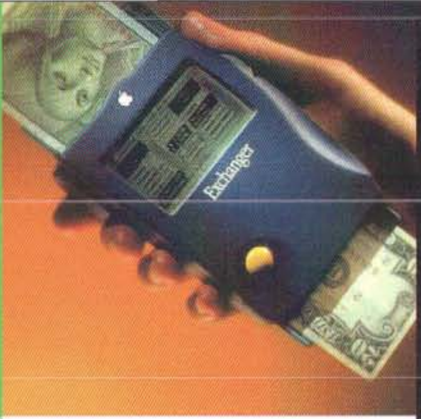
Although this article has provided a high-level examination of select elements defined by the CLI, the official documents are available in Microsoft Word format at http://msdn.microsoft.com/net/ecma/. If you would rather obtain the PDF versions of these same documents, navigate to http://www.ecma-international.org/publications/standards and locate the following two documents:

ECMA-334: The formal specification of the C# programming language.

32

< 1 year
2 years >

$64

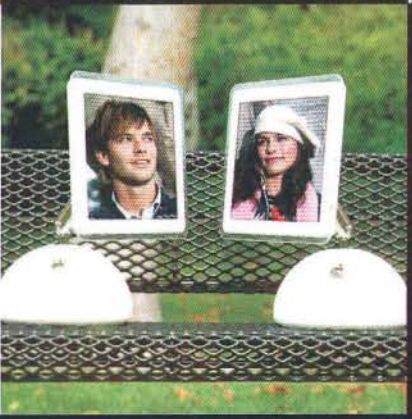


INTERVIEWS

TAPPING INTO THE WORLD OF CELEBRITIES AND THEIR MACS, ONLY MACDIRECTORY OFFERS EXCLUSIVE INTERVIEWS. GET A CLOSE AND PERSONAL VIEW FROM SARAH JESSICA PARKER, STING, STEVE JOBS, MADONNA, HARRY CONNICK JR., GEORGE LUCAS, JENNIFER JASON LEIGH, STEVE WOZ AND OTHER LEADERS IN THE MAC COMMUNITY.

FEATURES

DESIGNERS, WRITERS, MUSICIANS, BUSINESS LEADERS & OUR TECHNICAL EXPERT TEAM OFFER THEIR OWN PERSONAL INTERPRETATION OF THINGS THAT ONLY THE MAC SYSTEM CAN DELIVER. FEATURING OVER 240 PAGES OF REVIEWS, INTERVIEWS, NEWS, INSIGHTS, TRENDS AND THE LARGEST MACINTOSH BUYERS' GUIDE INCLUDING OVER 5,000 MAC PRODUCTS AND SERVICES.



CULTURE

MACDIRECTORY TAKES YOU TO THE WILDEST CORNERS OF THE WORLD AND UNCOVERS HOW MACINTOSH COMPUTERS ARE BEING USED BY OTHER CULTURES. TRAVEL TO JAPAN, AUSTRALIA, GERMANY, BRAZIL, INDIA, RUSSIA & LEARN MORE ABOUT APPLE'S CULTURAL IMPACT AROUND THE GLOBE.

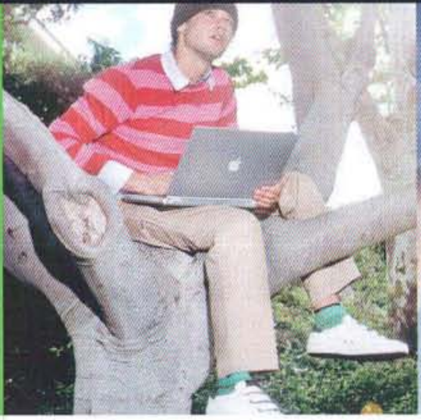
MacDirectory

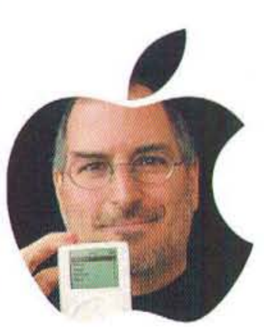
BEYOND ANY MACINTOSH MAGAZINE. SUBSCRIBE.

< Subscribe

ww.macdirectory.com/mw.html

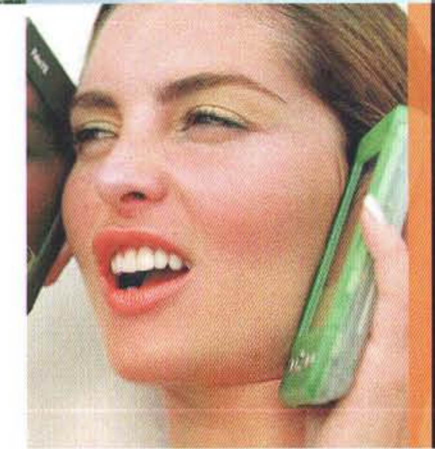
SEND CHECK OR MONEY ORDER TO:
MACDIRECTORY SUB DEPT.
326 A STREET, 2C
SOUTH BOSTON, MA 02110





REVIEWS

FIND OUT ALL YOU NEED TO KNOW ABOUT THE LATEST MAC PRODUCTS INCLUDING THE HOTTEST MAC OS SOFTWARE AND HARDWARE.



WIN!

SUBSCRIBE TO MACDIRECTORY AND YOU WILL AUTOMATICALLY ENTER OUR SWEEPSTAKES FOR A CHANCE TO WIN A NEW TITANIUM!

ECMA-335: The formal specification of the .NET CLI.

Yes, it's true. The Microsoft CLI documents have been submitted to ECMA as an official standard. In fact, the CLI was developed in conjunction with a number of key industry players (such as Intel and IBM).

ECMA-334 defines the syntax and semantics of C#. While I certainly encourage you to obtain and read this document, do be aware this is a rather academic treatment of the subject. Keep it handy for a rainy evening or a sleepless night, however do know that you will come to know the details of the C# programming language over the course of the articles to come.

ECMA-335 is a very critical document for all of those interested in understanding how the .NET platform can be ported to different operating systems. As well, this information is also quite helpful for tool builders who wish to build compilers for languages that target the .NET platform or geeks like us who are interested in technology.

Given that ECMA-335 is quite lengthy, it has been subdivided into a set of *partitions*, which is simply a fancy and overly verbose way of saying *sections*. **Table 4** explains the content found within these partition documents.

Partition of the ECMA-335 Document	**Meaning in Life**
Partition I Architecture	Describes the overall architecture of the CLI, including the rules of the CLS, the .NET type system (CTS), and the virtual execution system (VES).
Partition II Metadata	Describes how .NET types (and their members) are to be represented in terms of .NET metadata.
Partition III CIL	This document describes all of the gory details of the Common Intermediate Language (CIL).
Partition IV Library	Gives a high level overview of the minimal and complete class libraries that must be supported in a .NET distribution.
Partition V Annexes	A collection of 'odds and ends' that describe class library design guidelines, as well as the implementation of ilasm (the CIL assembler utility).

***Table 4.** A breakdown of the ECMA-335 specification.*

Always remember that at its core, the CLI is effectively little more than a set of white papers. All of the specifications set forth in these standards documents are of little practical use until they are given life in terms of a physical code base. To date, there are many implementations of the standards defined by the CLI. Some of these CLI distributions have indeed come from Microsoft proper, which as noted, are rather Windows-centric in that they leverage certain aspects of the Win32 operating system that are simply not found on other systems.

Lucky for us, there are a number of additional CLI distributions that have no ties to Microsoft (or the Win32 APIs) beyond the fact that the original specifications have been consulted as a foundation. Two of the most popular open source distributions of the CLI go by the names Portable .NET and Mono. **Table 5** describes the most well known distributions of the CLI.

.NET Platform Distributions	**Meaning in Life**
The Microsoft .NET Platform	This is the commercial CLI implementation offered by Microsoft, which is geared for use by the Windows family of operating systems.
The Microsoft .NET Compact Framework	A stripped down version of the official MS .NET platform that targets handheld computer devices (such as mobile phones and Pocket PCs).
The Microsoft Shared Source CLI (SSCLI) http://msdn.microsoft.com/net/sscli	While this implementation of the CLI has come from Microsoft, the code base is platform natural. This distribution is more geared towards R&D efforts and tinkering and is not intended to be a full-fledged development platform.
Portable .NET www.dotgnu.org	This is an open source CLI implementation that intends to account for all of the functionality found in the commercial Microsoft .NET platform (in addition to Unix-specific APIs).
Mono www.go-mono-org	Another open source CLI implementation that also intends to compete with the commercial Microsoft .NET platform (also in addition to Unix-specific APIs).

***Table 5.** Various distributions of the .NET Platform*

Again, given that you are reading MacTech (as opposed to Windows Developers Journal) I will not spend much time at all talking about the official Microsoft .NET distributions. Over the next several articles to come, we will come to know the SSCLI. As you will see, the SSCLI is a research-focused distribution of the CLI that is a great first step for Macintosh developers interested in checking out the .NET platform and the C# programming language.

While this particular CLI distribution does *not* provide an implementation of GUI based toolkits (such as Windows Forms or ASP.NET), it does provide a wealth of technical information that will build your .NET muscle. As you examine the SSCLI, you will become familiar with .NET namespaces found on any CLI distributions.

Later articles will build upon your exploration of the SSCLI by examining the Portable .NET and Mono open source projects. These CLI distributions not only provide an implementation of ADO.NET, ASP.NET and Windows Forms, but ship with compilers which target a number of .NET-aware programming languages. Unlike the SSCLI, Portable .NET and Mono are intended to be a full-blown framework for 'real-world' .NET application development.

Wrap Up

So! Here you are at the end of this article and you have yet to compile any code and undoubtedly have a number of unresolved questions regarding the .NET universe. Fear not, this was by design. The point of this first article was to lay the conceptual groundwork for the articles to come. First you learned that the CLI defines a *common* file format, *common* execution engine and *common* type system that can be implemented on the Mac OS X, Unix, Linux and Windows operating systems. Next you checked out some (but not all) of the namespaces provided by .NET, followed by an investigation of the language- and platform-agnostic aspects of the framework.

So, until next month, when we install and explore the SSCLI, allow me to welcome you to the .NET platform!

Macintosh

Bring your games, MP3's, or DVD's to Life with Professional Sound on Macintosh!

JBL Invader Speakers 4.1 Surround Sound!

The latest in sound innovation, these speakers enable full 4.1 surround sound multimedia system for a full rich sound field during music, movies and games.

$139.95

Griffin PowerWave!

DevDepot Price

$94.95

A powerful and extremely flexible computer audio tool, PowerWave can record any mic or line input into your computer. Use it as an external amp to connect any set of home speakers to your computer!

Macally Noise Reduction Head Phones!

DevDepot Price

$64.95

The perfect companion for a long flight, iPod, or anyone who wants the best listening experience! Cancels out background sound! Includes an air line adapter to plug directly into most airplanes.

iTrip for iPods

Now for New iPods!

DevDepot Price

$34.95

Let's you play your music through any FM radio -- in your car, at a party, wherever! Needs no batteries! Programmed from your iPod and now available it two versions, for all iPods!

Toll Free: 877-337-6866 • Outside US/Canada: 805-494-9797 • Fax: 805-494-9798 • orders@devdepot.com
PO Box 5200 • Westlake Village, CA 91359-5200

By Clark Jackson, Tacoma Power

Printing One Page Reports

How to accomplish simple program controlled printing in Cocoa

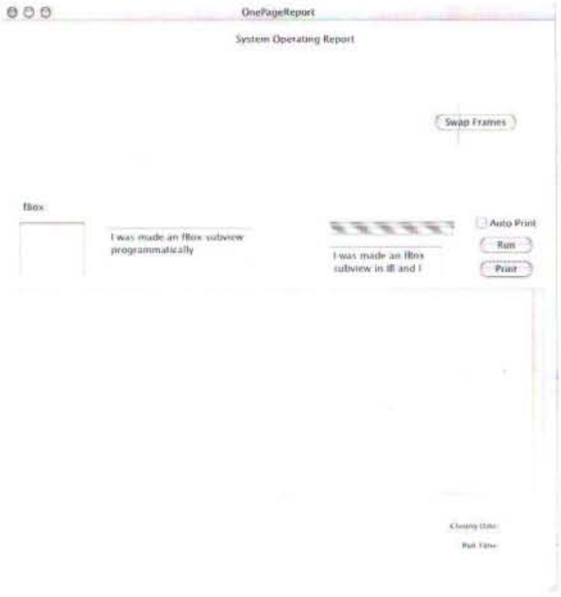
This is another article directed at the enterprise. The enterprise because it is there that you often find the need for one page reporting of many stripes. Many enterprise reports have data from disparate sources scattered over the page but yet are not complex enough to demand an NSDocument based application. Using Interface Builder makes the layout of one page reports easy so all is well–that is, until you need to print that report. Yes, you can tell any NSView to print itself but that doesn't help too much if the views print individually. And, what if you want to bypass the print dialog and have the output scaled and the orientation landscape? This tip is meant to provide the needed information to print simple reports where page breaking is not involved.

Topics

- Managing view hierarchy
- Collecting views for printing
- Benefits of subclassing NSWindowController
- Anatomy of frames
- Moving and resizing views
- Specifying margins, orientation, scaling, paper, and copies
- Bypassing the print panel
- Mid-code variable declarations
- Argument-passing timers

Arranging Views

The window of our sample program is presented in Figure 1. It contains information we want to print and information we don't want to print including various UI elements. Its orientation is portrait but the information we want printed is better suited to landscape and it's too big to fit on one printed page. The Page Setup... command allows us to control the scale but when we adjust it to fill the page, the default too-wide margins prevent us from doing so. Finally, we'd like the report to print just after midnight when no users are present.



I was made an fBox subview programmatically

I was made an fBox subview in IB and I

***Figure 1.** The window containing our report.*

The simple approach to one page printing is to tell the window to print itself: [[someReportUIElement window]print:nil]; where someReportUIElement is any user interface element that is in the report window. The nice thing here is that windows seem to scale and orient themselves automatically to fit a page the best way. The drawback here is that a window printing itself includes its title bar and the window's background horizontal gray pinstripes. The pinstriping can be handled permanently or during a print by the following:

```
[[someReportUIElement window]setBackgroundColor:
[NSColor whiteColor]];
[[someReportUIElement window] print:nil];
[[someReportUIElement window]setBackgroundColor:
[NSColor windowBackgroundColor]]; // restores pinstripe
```

If he had to do it all over again **Clark** would choose to be born one of the Sons of Liberty. The fact that the main Boston organizer was Ebenezer *McIntosh* and that many of the group were printers and publishers is not lost on him. He can be contacted at cjackson@cityoftacoma.org.

Another way to easily print a one page report would be to tell the window to print its *contents* thus eliminating the title bar and gray pinstripe: [[[someReportUIElement window]contentView]print:nil];. The drawback here is that the scaling and orientation don't calculate automatically and our other objectives remain unmet.

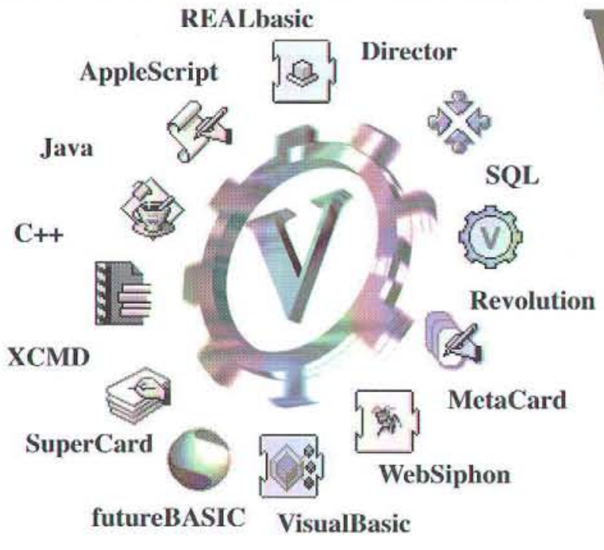
It should be noted here that if your controller is subclassed from an NSWindowController then the print statement can be: [[self window]print:nil]; or [[[self window] contentView]print:nil]; (remember to make the connection in IB between the window and the controller's "window" outlet!). Subclassing NSWindowController has the added benefit of having windowWillClose: being called on it automatically (by making the controller the window's delegate in IB) so that you can release your resources when the window is closed. Releasing your controller is not an issue for simple apps that only have one controller; however, with more complicated applications with many controllers that come and go, windowWillClose: is one way to be notified when your window's controller can be released.

The most flexible solution to printing views together is to provide a faceless background superview and tell it to print itself including its subviews. A likely candidate view for this purpose is an NSBox. Start by dragging an NSBox onto our window in IB. Make it big enough to cover the area you want printed. A custom view in IB would serve as well but the NSBox has the added ability to draw a border and a title if you should want them. The inspector in IB doesn't give you the options to specify where the title appears but you can do it programmatically. Possible constants are NSNoTitle, NSAboveTop, NSAtTop, NSBelowTop, NSAboveBottom, NSAtBottom, and NSBelowBottom.

Any element you want to print along with your NSBox view has to be a subview of that NSBox view. You can assign UI elements to be subviews of the NSBox view either in IB or programmatically. In order to assign them in IB, drag your NSBox view onto the window *first*. Then drag the other elements you want printed *from the palette* on top of the NSBox (you will see the NSBox view highlight).

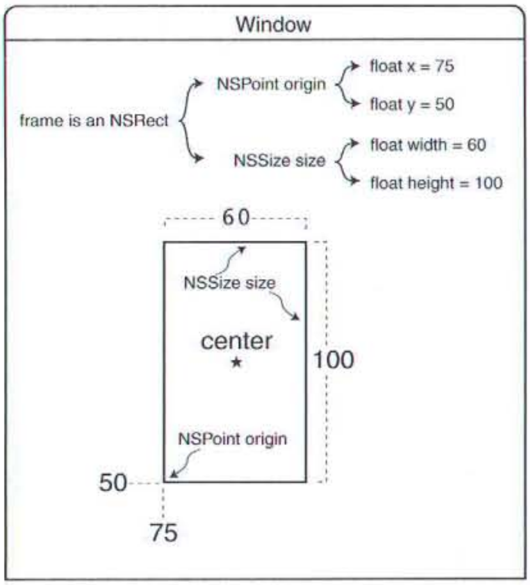
If you choose to assign elements programmatically it takes a little more work because you have to assign a new frame location. Let's say you place an NSBox view on your window after placing an NSTextField. You send the NSBox view to the back and put the NSTextField on top. The NSBox view doesn't highlight as you drag NSTextField on top of it because the NSTextField hasn't come directly off the palette. As a result, the NSTextField does not become a subview of the NSBox view. To fix this situation in your program you would make the NSTextField (fNotSubviewTextField) a subview of your NSBox (fBox), in this way: [[fBox contentView] addSubview:fNotSubviewTextField];. Unfortunately, our work is not done because fNotSubviewTextField keeps its frame attributes from its previous superview (the window) and applies them to the new superview (the fBox) most likely causing fNotSubviewTextField to disappear by being outside the clipping area of fBox. (By the way, variables starting with "f" indicate an instance variable, a holdover from my old MacApp days.) Preserve fNotSubviewTextField's location (so it is not clipped) relative to the *window* in this way:

```
NSRect originalTextFieldFrame = [fNotSubviewTextField frame];
// get the original frame based on the window being the superview
[[fBox contentView] addSubview: fNotSubviewTextField]; // move
the text field to the box view for printing
```



```
NSRect newTextFieldFrame = originalTextFieldFrame; // copy
original frame into new, later to change origin not size
// make allowance for the NSBox's border
float xAdj = 0.0;
float yAdj = 0.0;
if([fBox borderType] == NSLineBorder) xAdj = yAdj = 1.0;
else if([fBox borderType] == NSBezelBorder || [fBox
borderType] == NSGrooveBorder) xAdj = yAdj = 2.0;
boxFrame = [fBox frame]; // get the new superview's frame
// calculate the new frame using the difference between the original and new
superview frames
newTextFieldFrame.origin.x = originalTextFieldFrame.origin.x
- boxFrame.origin.x - xAdj;
newTextFieldFrame.origin.y = originalTextFieldFrame.origin.y
- boxFrame.origin.y - yAdj;
[fNotSubviewTextField setFrame:newTextFieldFrame]; // give the text
field its new frame in terms of it's new superview
```

The frame method of an NSView returns an NSRect structure that defines its position in its superview. For those of you new to Cocoa, not all names preceded by "NS" refer to Objective C objects, some like NSRect, NSSize, NSPoint, and NSRange are C structures and therefore have elements that are accessible via the . syntax, i.e. NSPoint center.x = [fBox frame].size.width / 2.0; works just fine. Figure 2 illustrates the hierarchy.



The coordinates of the center point of the box

in terms of its own coordinate system:
center.x = [fBox frame].size.width/2.0
center.y = [fBox frame].size.height/2.0

in terms of the window's coordinate system:
center.x = [fBox frame].origen.x
+ [fBox frame].size.width/2.0
center.y = [fBox frame].origen.y
+ [fBox frame].size.height/2.0

Figure 2. *The anatomy of a view's frame.*

Now fNotSubviewTextField will print (inspite of its name!), having programmatically become a subview of fBox, when fBox is told to print. The next problem to resolve is subviews of fBox that you *don't* want to print. Figure 1 shows a few elements inside of fBox that we don't want to print: fRunButton, fPrintButton, and fProgressIndicator. Notice we do not include the fAuto check box in this list because even though it appears on top of fBox it is not a subview of fBox and therefore will not print with fBox. Until Panther ships, which adds the ability to hide NSViews, we will have to programmatically move unwanted views outside the clipping bounds of fBox before printing—and put them back after.

In order to move our views around conveniently we'll use a two step process. First, we'll set up the off-view set of frames one time when our program launches and second, we'll provide a method that swaps the frames back and forth. We'll need an instance variable array of the UI element frames, fRelocatableFrames. When awakeWithNib is called we specify the NSRect's that are initially the off-view frames for fBox's subviews that we don't want to print. Since NSRect's are not objects we'll need to reference them in the array by index so we enumerate an index as well. The final thing we'll need is an array of the affected UI elements, fRelocatableObjects. This array will be used in the method that swaps the frames of the objects.

```
// make a list of all the views that you want relocated, resized, or hidden during
printing
typedef enum
{ kRunButton,
  kPrintButton,
  fProgressIndicator,
  kRelocateTextField
} ElementsToHideWhilePrinting;

// populate fRelocatableFrames so designated user interface elements can be hidden
or relocated during printing
NSSize myOffViewSize;
NSPoint myOffViewLocation;
myOffViewLocation.x = 1700.0; // an arbitrary off-view location
myOffViewLocation.y = 1700.0;
fRelocatableFrames[kRunButton].origin = myOffViewLocation; //
remember off-view location
fRelocatableFrames[kRunButton].size = [fTextView frame].size;
// remember original size
...
// fTextView will be different from the others in that we still want it to print but at a
different location and size
  myOffViewSize.height = 65.0;
  myOffViewSize.width = 200.0;
  myOffViewLocation.x = 320.0;
  myOffViewLocation.y = [fBox frame].size.height -
myOffViewSize.height - 20.0;
  fRelocatableFrames[kRelocateTextField].origin =
myOffViewLocation;
  fRelocatableFrames[kRelocateTextField].size = myOffViewSize;

  // now that the new frames have been created, make a list of affected UI objects
  // so we can iterate over them swapping frames as we go
 fRelocatableObjects = [NSMutableArray arrayWithCapacity:5];
[fRelocatableObjects retain];
[fRelocatableObjects insertObject:fRunButton
atIndex:kRunButton];
...
```

During program execution we need a method that will assign the new frames to the relocatable objects at the same time remembering the original locations and sizes so that they can be restored after printing:

```
                                                          swapFrames
This method conveniently handles the moving and resizing of any element during
printing. It remembers the old location so the pre-printing state can be restored.


- (IBAction)swapFrames:(id)sender
{
  int theIndex, theNumberOfObjects;

  theNumberOfObjects = [fRelocatableObjects count];
  {
    // Why the brace? arrayOfNewFrames is declared below as an NSRect only after
theNumberOfObjects has been determined. Declaring new variables has to be done
inside code blocks i.e. inside braces, {}
    NSRect arrayOfNewFrames[theNumberOfObjects];

    // make a copy of the relocateble frames
    for(theIndex = 0;theIndex <
theNumberOfObjects;theIndex++)
    {
      arrayOfNewFrames[theIndex] =
fRelocatableFrames[theIndex];
    }

    // put the existing frames of the relocatable objects into the fRelocatableFrames array,
these frames will be remembered here so that they can be swapped back in the future
    for(theIndex = 0;theIndex <
theNumberOfObjects;theIndex++)
    {
      fRelocatableFrames[theIndex] = [[fRelocatableObjects
objectAtIndex:theIndex]frame];
    }

    // now impose the new set of frames on the objects to be relocated/resized
    for(theIndex = 0;theIndex <
theNumberOfObjects;theIndex++)
    {
      // to remove any vestige of the UI element once it has been moved
      [[[fRelocatableObjects objectAtIndex:theIndex]
superview] setNeedsDisplayInRect:[[fRelocatableObjects
objectAtIndex:theIndex] frame]];
      // assign the new frame
      [[fRelocatableObjects objectAtIndex:theIndex]
setFrame:arrayOfNewFrames[theIndex]];
      // make sure it redraws itself after being moved
      [[fRelocatableObjects objectAtIndex:theIndex]
setNeedsDisplay:YES];
    }
  }
  return;
}
```

The source includes another method, moveForm, that demonstrates moving an NSForm into fBox (making it an fBox subview) during printing and back out into the window again afterwards. By modifying the statements in these three methods, awakeFromNib, swapFrames, and moveForm you should be able to move, resize, hide, print, and afterwards

It's 3AM on Sunday morning...
Available at DEPOT www.devdepot.com
IS THE SERVER STILL RUNNING?
End the worry with
Kick-off!®
The simple reliability solution for Mac OS 9 and X
System-level Crash Detection
patented hardware-software integration
Automatic Crash Recovery
system restart or power cycle, as needed
Monitor Custom Applications
software developer's kit and plug-ins included
Scheduled Restarts
cures memory leaks and fragmentation
Restart after Power Failure
even if shut down by your UPS
Comprehensive Error Logging
to help you track down problems
Simple Installation
just plug in AC cord and USB cable
For all these features plus six power outlets controllable by phone tones, schedules and scripts, consider our PowerKey Pro 650 Admin.
www.sophisticated.com
SOPHISTICATED CIRCUITS INC
Copyright ©2002 Sophisticated Circuits, Inc. Kick-off! and PowerKey are registered trademarks of Sophisticated Circuits, Inc. Mac OS is a registered trademark of Apple Computer, Inc.

restore any number of views that you have to meet any single page printing requirement.

Basic Printing

Once all our views are in place to print (or *not* print) we need a method that will direct fBox to print. fBox's output, our report, appears in Figure 3. The method in our program defaults to *not* using a print dialog, however, that ability remains at the user's discretion by using the option key. In the default no-user-interaction mode our method will specify page orientation, scale, margins, and number of copies. In order to print bypassing the user we instantiate an NSPrintInfo which contains all the print settings we need including margins and page orientation. You can choose to have the scaling be automatic to fit the page by using setHorizontalPagination:NSFitPagination or you can get the NSPrintInfo's dictionary and set the scaling directly. With that same dictionary you can specify the number of copies. Following is the source that encapsulates what you need for printing.

```
myPrintInfo = [[NSPrintInfo alloc]
initWithDictionary:(NSMutableDictionary*)[[NSPrintInfo
sharedPrintInfo]dictionary]];  // get a copy of the shared NSPrintInfo
provided by the system

// adjust the margins
[myPrintInfo setOrientation:NSLandscapeOrientation]; // alt:
NSPortraitOrientation
[myPrintInfo setBottomMargin:30.0];
[myPrintInfo setLeftMargin:30.0];
[myPrintInfo setRightMargin:30.0];
[myPrintInfo setTopMargin:35.0];

//You can specify the paper name here, just make sure your printer has it for
unattended printing
// [myPrintInfo setPaperName:@"Legal"];
// you can have scaling to be automatic here or set the scaling factor as shown below
// [myPrintInfo setHorizontalPagination:NSFitPagination];
// [myPrintInfo setVerticalPagination:NSFitPagination];

// set up the dictionary, get it from your NSPrintInfo
myPrintInfoDictionary = (NSMutableDictionary*)[myPrintInfo
dictionary];
[myPrintInfoDictionary setObject:[NSNumber
numberWithFloat:0.65] forKey:NSPrintScalingFactor];
[myPrintInfoDictionary setObject:[NSNumber numberWithInt:1]
forKey:NSPrintCopies];

// Use either of these statements below to print the window or its contents respectively
//myPrintOperation = [NSPrintOperation printOperationWithView:[self window]
printInfo:myPrintInfo];
//myPrintOperation = [NSPrintOperation printOperationWithView:[[self window]
contentView] printInfo:myPrintInfo];

// run your print job on fBox
myPrintOperation = [NSPrintOperation
printOperationWithView:fBox printInfo:myPrintInfo];
[myPrintOperation setCanSpawnSeparateThread:YES];
[myPrintOperation setShowPanels:NO]; // don't want to see the panel
[myPrintOperation runOperation];

[myPrintInfo release]; // it was alloc'd so release it
```

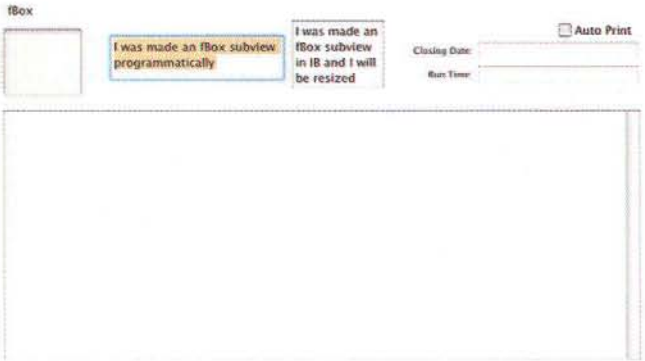

fBox
I was made an fBox subview programmatically
I was made an fBox subview in IB and I will be resized
Closing Date
Run Time
□ Auto Print

Figure 3. *fBox as printed, constituting our one page report.*

Unattended Printing

The final thing to accomplish is to provide a means to print the report unattended. We accomplish this by using an NSTimer. As soon as you introduce a timer you need to think about the method it will be calling or invoking. If it is necessary to pass any parameters to your printing method from your timer then you will need to set up a printing method different from the one provided by IB that is linked to your Print button. The source demonstrates segregating printing functions by using **printUnattendedWithScaling:andCopies:**. This method has two parameters which the timer will provide at the time of unattended printing. It is also called by the UI Print button (with nil arguments) when the user selects to print without a print panel. Using an NSTimer is shown in the following method from the source:

createPrintTimer:

This method creates and releases a timer (when the user toggles the switch) that controls unattended printing at midnight. If you are new to NSTimer an interesting aspect is how arguments are passed to the method that is invoked by the timer. Also, creating and disposing of NSTimer's is shown. This timer is set to fire every 30 minutes. The printUnattendedWithScaling:andCopies: method does the checking to verify the time and whether or not the report has already printed once for the day.

```
- (void) createPrintTimer:(id)sender
{
  NSInvocation *printUnattendedInvocation;
  SEL theSelector;
  NSMethodSignature *aSignature;
  NSNumber *myTwo,*my65Percent;

  // these will be passed as arguments, arguments must be objects
  myTwo = [NSNumber numberWithInt:2];
  my65Percent = [NSNumber numberWithFloat:0.65];

  if(fPrintTimer) // timer already exists so dispose of it
  {
    if([fPrintTimer isValid])
    {
      fPrintTimer invalidate];
      [fPrintTimer release];
      fPrintTimer = nil;
    }
    else
    {
      NSLog(@"should never end up here where timer exists
and is invalid");
```

```
      }
   }
   else // timer doesn't exit, create timer
   {
      // include line below if you want the method called as soon as the timer is
turned on, timers fire first time AFTER period has passed
      // [self printUnattendedWithScaling:my65Percent andCopies:myTwo];
      theSelector = @selector(printUnattendedWithScaling:
andCopies:);
      aSignature = [MyWindowController
instanceMethodSignatureForSelector:theSelector];
      printUnattendedInvocation = [NSInvocation
invocationWithMethodSignature:aSignature];
      [printUnattendedInvocation setSelector:theSelector];
      [printUnattendedInvocation setTarget:self];
      [printUnattendedInvocation setArgument:&my65Percent
atIndex:2]; // index 2 is where arguments to the method begin, note ampersand
      [printUnattendedInvocation setArgument:&myTwo atIndex:3];
      fPrintTimer = [[NSTimer
scheduledTimerWithTimeInterval:60*30
invocation:printUnattendedInvocation repeats:YES]retain];
// 60*30 is timer repeat period, (seconds/minute)*minutes
   }
}
```

CONCLUSION

We have resolved many of the single page report printing issues. For example, we have answered how to collect views for printing making subviews both in IB and programmatically. We have show how to relocate and resize views in a clean way for printing including the ability to exclude views from output. We have shown how to bypass the print panel specifying number of copies, orientation, paper name, scaling, and margins. Finally, we constructed a simple argument-passing timer that will run your print jobs at any specified time. For multi-page printing jobs there is always NSDocument.

HELP AUTHORING

Mac OS 9 Mac OS X
Help
Windows Linux

By MacTech Reviews

Application Help

Authoring and Deploying Help in a Multi-Platform World

Introduction

Every significant desktop application requires an integrated help system. Most successful applications will ultimately run on Mac OS X, Windows, and possibly other platforms like Mac OS 9, and Linux. Product development is seldom a one-time event, but rather an ongoing process of enhancement, evolution, and bug fixing. Developers need a strategy for authoring, deploying, and maintaining help systems in a multiple platform world.

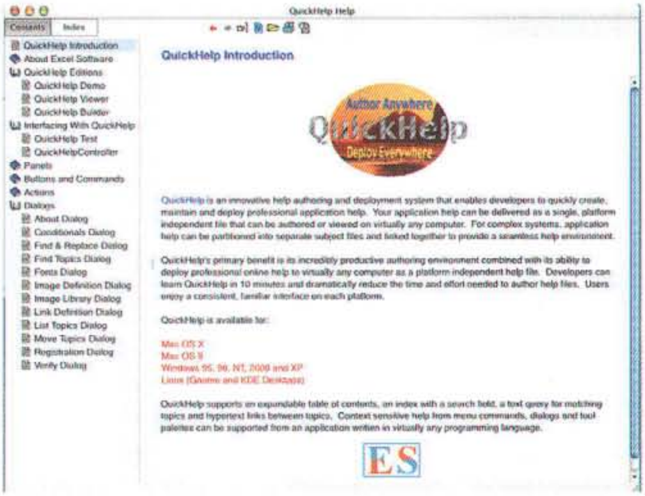
Some IDE (Interactive Development Environment) vendors have recognized the need to develop for multiple platforms. With RealBasic, from Real Software, you can create your application once with a bit of conditional logic to handle platform-specific issues, then generate builds targeted for Mac OS X, Mac OS 9 and/or Windows. Likewise, Borland's Delphi is a popular IDE for Windows that has a source code compatible cousin on Linux named Kylix. These tools enable your application to take advantage of the unique features, and user interface, of each operating system, yet can substantially reduce the cost of multi-platform development.

MacHelp is the HTML based help format developed by Apple for Macintosh software. Most Windows applications use Microsoft's HTML Help or its predecessor, WinHelp. Many help authoring tools have been developed around these formats. Unfortunately, when Apple and Microsoft created their platform specific help formats, easy porting to competing platforms was not a high priority.

Linux started as a variant of Unix, with a command line interface, but in recent years improvements to the KDE and GNOME desktop environments have made Macintosh and Windows users feel right at home. Help information for Linux has traditionally been delivered as ReadMe files, MAN pages, or HTML files. These approaches are awkward and inadequate for new GUI programs arriving on Linux as it moves to the desktop in more organizations.

Help authoring approaches for Macintosh typically rely on editing a collection of topic files using HTML tags for formatting. Lots of time is spent on formatting pages, organizing topics and managing and maintaining hundreds of links between topics. Given enough time and effort, you'll end up with a nice help system for your Mac OS X application, but then must substantially rework it when porting to Windows or Linux.

QuickHelp is a new authoring tool based on the idea of writing help once, then deploying it to virtually any platform including Mac OS 9, Mac OS X, Windows 95 to XP and Linux KDE or GNOME. This article provides an overview of QuickHelp, shows how to develop a help system, add context sensitive application links to help topics, and then deploy your help system to Macintosh, Windows, and Linux computers.



QuickHelp Introduction

QuickHelp is an innovative help authoring and deployment system that enables developers to quickly create, maintain and deploy professional application help. Your application help can be delivered as a single, platform independent file that can be authored or viewed on virtually any computer. For complex systems, application help can be partitioned into separate subject files and linked together to provide a seamless help environment.

QuickHelp's primary benefit is its incredibly productive authoring environment combined with its ability to deploy professional online help to virtually any computer as a platform independent help file. Developers can learn QuickHelp in 10 minutes and dramatically reduce the time and effort needed to author help files. Users enjoy a consistent, familiar interface on each platform.

QuickHelp is available for:

Mac OS X
Mac OS 9
Windows 95, 98, NT, 2000 and XP
Linux (Gnome and KDE Desktops)

QuickHelp supports an expandable table of contents, an index with a search field, a text query for matching topics and hypertext links between topics. Context sensitive help from menu commands, dialogs and tool palettes can be supported from an application written in virtually any programming language.

QuickHelp Viewer on Mac OS X

QuickHelp Overview

Most help systems, including MacHelp, store topics as a folder of HTML pages that reference other files containing images and indexes. Microsoft's HTML Help lets you compile your help system to a compressed file, making it easier to deploy. When users activate help for an application, a separate viewer executable allows them to navigate and view topics. The help format and Viewer executable are tied to a specific platform.

QuickHelp stores the topic text, images and organizational information for your help system in a platform neutral XML file. XML (eXtensible Markup Language) is an industry standard format for storing and exchanging information. It describes information as tagged data in a text file that's human readable, self-documenting, and widely supported throughout the software industry by editing and parsing tools.

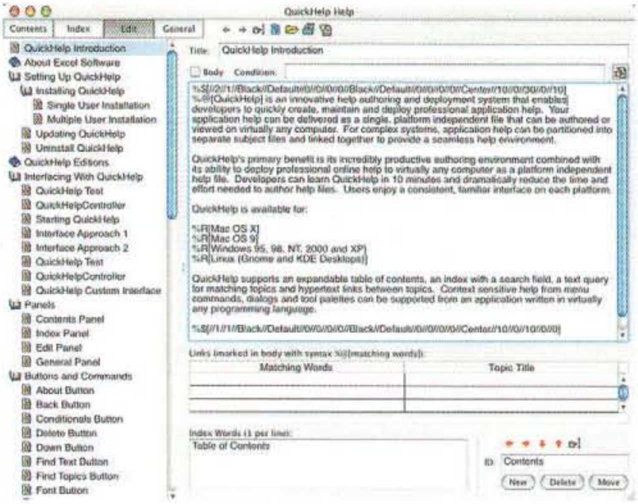
Like any help system, topics and navigational features are presented via a Viewer. There are four QuickHelp Viewers, one each for Mac OS 9, Mac OS X, Windows and Linux. The Windows Viewer runs on any Windows operating system from 95 through XP. The Linux Viewer runs on all popular Linux distributions including Mandrake, Red Hat and Suse, and supports the leading desktop environments KDE and GNOME. QuickHelp Viewers for MacOS X and Windows can be downloaded from shareware sites. QuickHelp developers can distribute the Viewer executable royalty-free along with their help file.

Regardless of the tools you use, help authoring can be characterized as an iterative Organize-Edit-Build-Test process. Most authoring tools give you a project environment where you can add and organize HTML files and images. Some tools let you edit topics directly while others require separate text or HTML editors. Creating topics with colored text, formatted images, and hypertext links can be a labor-intensive process when using HTML tags. Most tools use a build process that links everything together before you can view and test the finished help system. Depending on the authoring tool, there may be features for locating missing links and image references.

The Builder edition of QuickHelp tightly integrates the Organize-Edit-Build-Test process into a single tabbed window. The Builder looks just like the Viewer, with the addition of the Edit and General panels for organizing, editing, and managing your help file. From the Edit panel, you can create, edit, and delete topics and organize them in the Contents tree. Your help information is stored directly in the XML file. When you're ready to view the finished help, just click the Contents or Index tab. QuickHelp eliminates the build process, and integrates the Viewer's features directly into the Builder for quick testing. By integrating the entire authoring process into one tabbed window, the learning curve for QuickHelp is reduced to about 10 minutes, and developer productivity is improved.

Creating Topics

The Edit panel is used to create topics in a help file. The left side of the panel has a table of contents that shows the title, indentation, and position of each topic. When a topic is selected, its information is shown in the fields on the right side of the Edit panel. There are fields to title the topic, assign conditions that affect its visibility, write its body text, identify index words that appear in the user's index, and assign a unique integer or string to identify the topic for an application's context sensitive help.



QuickHelp Introduction
About Excel Software
Setting Up QuickHelp
Installing QuickHelp
Single User Installation
Multiple User Installation
Updating QuickHelp
Uninstall QuickHelp
QuickHelp Editions
Interfacing With QuickHelp
QuickHelp Test
QuickHelpController
Starting QuickHelp
Interface Approach 1
Interface Approach 2
QuickHelp Test
QuickHelpController
QuickHelp Custom Interface
Panels
Contents Panel
Index Panel
Edit Panel
General Panel
Buttons and Commands
About Button
Back Button
Conditionals Button
Delete Button
Down Button
Find Text Button
Find Topics Button
Font Button

Title: QuickHelp Introduction

□ Body Condition:

```
%$[//2//1//Black//Default//0//0//0//0//Black//Default//0//0//0//0//Center//10//0//30//0//10]
%@[QuickHelp] is an innovative help authoring and deployment system that enables
developers to quickly create, maintain and deploy professional application help. Your
application help can be delivered as a single, platform independent file that can be authored or
viewed on virtually any computer. For complex systems, application help can be partitioned into
separate subject files and linked together to provide a seamless help environment.

QuickHelp's primary benefit is its incredibly productive authoring environment combined with
its ability to deploy professional online help to virtually any computer as a platform independent
help file. Developers can learn QuickHelp in 10 minutes and dramatically reduce the time and
effort needed to author help files. Users enjoy a consistent, familiar interface on each platform.

QuickHelp is available for:

%R[Mac OS X]
%R[Mac OS 9]
%R[Windows 95, 98, NT, 2000 and XP]
%R[Linux (Gnome and KDE Desktops)]

QuickHelp supports an expandable table of contents, an index with a search field, a text query
for matching topics and hypertext links between topics. Context sensitive help from menu
commands, dialogs and tool palettes can be supported from an application written in virtually
any programming language.

%$[//1//1//Black//Default//0//0//0//0//Black//Default//0//0//0//0//Center//10//0//10//0//0]
```

Links (marked in body with syntax %@[matching words]):

Matching Words	Topic Title

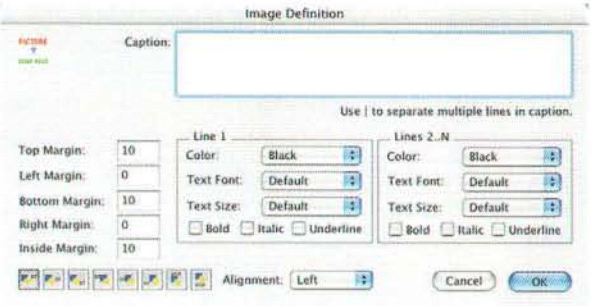
Index Words (1 per line):
Table of Contents

Edit Panel of QuickHelp Builder

The New button is used to create a topic in the table of contents below the selected topic. The Delete button deletes the selected topic. The Move button is used to move a single topic or group of topics to a different location or indentation within the table of contents. Use the arrow buttons to change the position or indentation of a selected topic.

Formatting Topics and Adding Images

The Insert Tag button, at the top right of the Body field, is used to insert a tag into the text of a topic for applying text styles, adding links to other topics, or inserting formatted images. Although it is possible to edit formatting tags directly in the Body field, it is easier to just click the Insert Tag button, and choose a popup menu command. For image tags, a dialog is presented to select an image and define caption text and formatting.



Caption:

Use | to separate multiple lines in caption.

Top Margin: 10
Left Margin: 0
Bottom Margin: 10
Right Margin: 0
Inside Margin: 10

Line 1
Color: Black
Text Font: Default
Text Size: Default
□ Bold □ Italic □ Underline

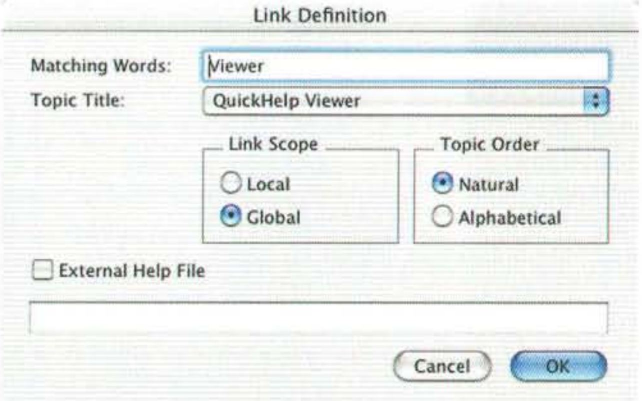
Lines 2..N
Color: Black
Text Font: Default
Text Size: Default
□ Bold □ Italic □ Underline

Alignment: Left

Image Definition Dialog

Linking Topics

Hypertext links connect highlighted words to other topics in the same help file, or in a different help file. Local and global link tables relate matching words in a topic body to the target topic. QuickHelp automates the process of creating and maintaining links when repositioning or renaming topics. To create a link, just select some text in the topic, click the Insert Tag button, and pick the target of that link from the Link Definition dialog.



Matching Words: Viewer
Topic Title: QuickHelp Viewer

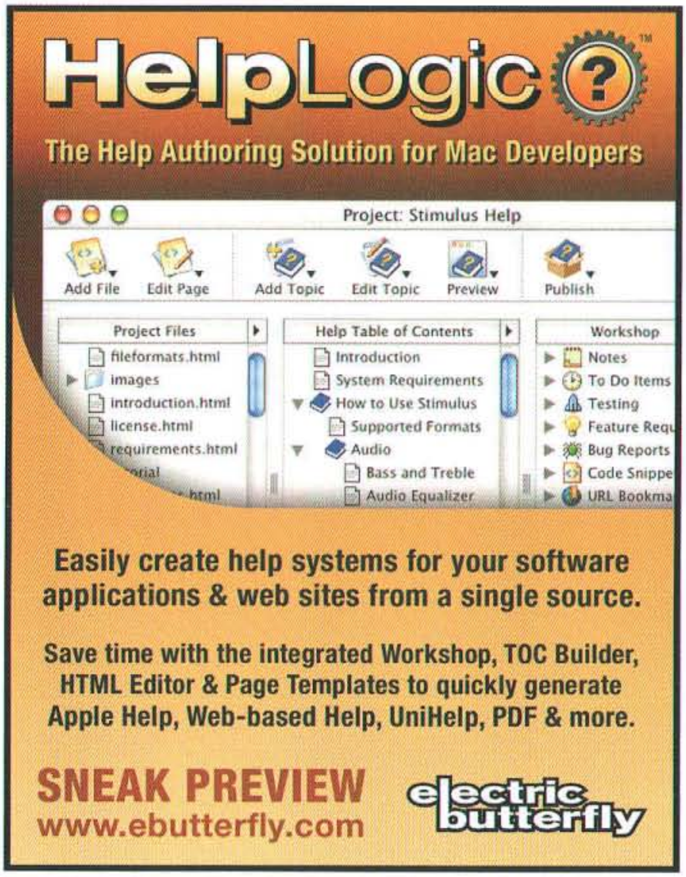
Link Scope
○ Local
● Global

Topic Order
● Natural
○ Alphabetical

□ External Help File

Link Selected Text from One Topic to Another Topic

Like HTML approaches, topics in QuickHelp are created with text and formatting tags. Topic editing in QuickHelp trades



some of the flexibility, and most of the complexity of creating HTML pages, for a simpler, more productive environment. The end result is professional looking topics that are consistently formatted and viewable on any platform.

Making Topics Conditional

Regardless of how similar your application is on each platform, there's bound to be a few platform specific features. You could create your help file for, say, the Macintosh platform, and then copy that file and modify it a bit to create a Windows version. While that's a lot less work than learning to use different help tools and rewriting the help system for each platform, you'll still have different help files to maintain for each new release of your application. Furthermore, larger applications are sometimes offered in several different editions on each platform, so you may want a customized help system for each edition. Using conditional topics, QuickHelp makes it easy to support multiple platforms and multiple editions of your application with a single help file.

For example, lets assume you have an application that's available in Demo, Standard and Professional editions, and it runs on Mac OS 9, Mac OS X, Windows, and Linux. During runtime, you'll probably want to present 12 slightly different flavors of your help system originating from one QuickHelp file.

Help topics can be conditional, and text or images within a topic can be conditional. When viewed in the Contents or Index panel of the QuickHelp Viewer, those topics, or portions of a topic, will be hidden or visible based on the current conditions.

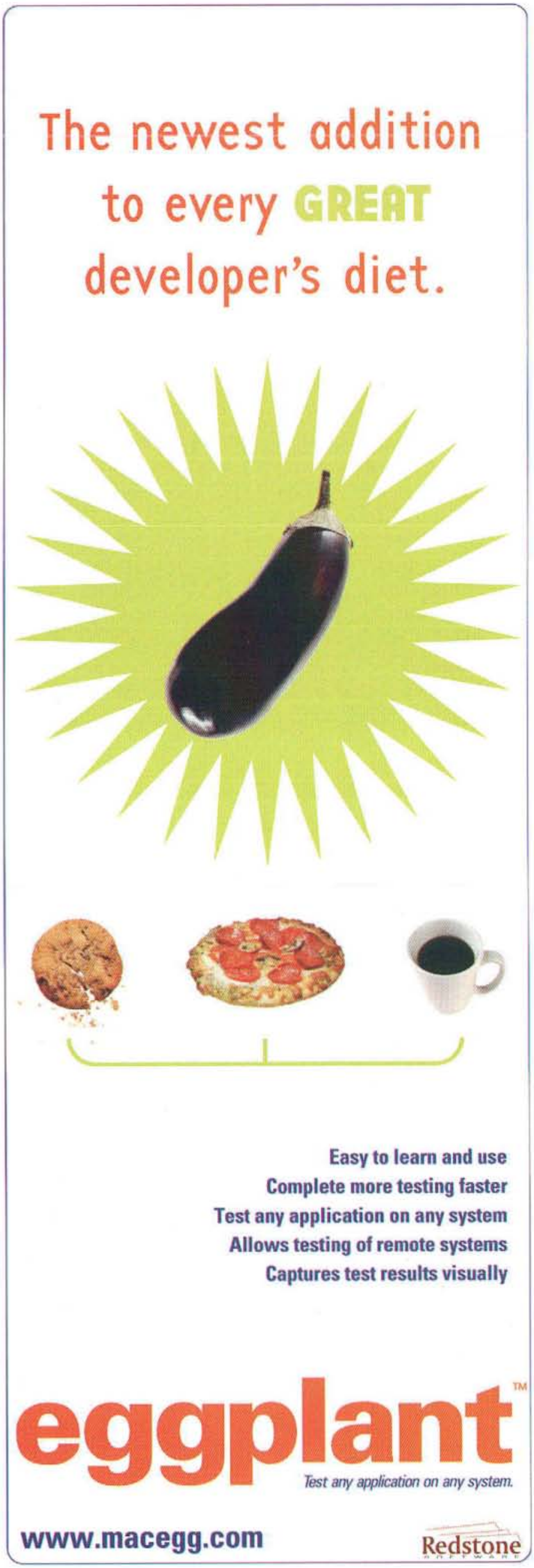
To conditionally include a topic, define its condition in the Condition field of the Edit panel. The conditional expression contains names, and Boolean operations that evaluate TRUE or FALSE during runtime. Use the Conditionals dialog, accessible from the General tab, to define names and/or set the current state of those names. In this example, we'll use the names Demo, Standard, and Professional.

Within the condition you can use AND, OR, XOR, NOT, (or) to create the expression. Here is an example that conditionally includes the topic in just the Professional edition when running on the Mac OS X platform:

```
Professional AND MacOSX
```

If the State checkbox is set for the Professional name in the Conditionals dialog, and this help file is currently running on Mac OS X, then the topic is visible, otherwise it is hidden.

Topics can be nested in a tree structure. If a topic is hidden, all topics nested under it are also hidden. It isn't necessary to repeat the conditions of any parent topic of a subtree, but it is okay to do so if it helps you remember exactly what conditions apply to each topic. QuickHelp automatically removes hyperlinks from the rest of your help system to those hidden topics during runtime.


The newest addition
to every GREAT
developer's diet.
Easy to learn and use
Complete more testing faster
Test any application on any system
Allows testing of remote systems
Captures test results visually
eggplant™
Test any application on any system.
www.macegg.com
Redstone SOFTWARE

To conditionally include a portion of text within the body of a topic, enclose it within #ifcondition, and #endcondition as shown below.

```
#ifcondition Professional AND MacOSX
You're running the Professional edition on Mac OS X
#endcondition
```

Locating Topics

The Index Words field of the Edit panel is used to automatically build an alphabetical index of keywords that the help author wants to associate with each topic. During runtime, the help user can locate, and navigate to topics from the Contents panel, or the Index panel by clicking hyperlinks within a topic, or by clicking on the Back and Forward buttons to step through recent viewing history.

The entire help system is usually deployed as a single XML file. For really large projects with different subject areas, you can chain separate help files together with hypertext links.

The help user can use the Find Topics dialog to locate help topics that contain specific words in its body.

Each Topic has an ID field that contains a unique number, or string, that can be used to identify that topic from the application when implementing context sensitive help. For example, if the user positions the mouse over a menu command, window, or dialog within your application, you can run the QuickHelp Viewer by having the user press F1, triggering the Viewer to present the related help topic. Context sensitive help is discussed below.

Working with Help Files

The General panel is used to read and write help files, edit global link definitions, and set preferences. There are command buttons to create, open, save, and verify help files. Help topics can be listed to text, or imported from text. The help developer can assign a default font which can later be changed by the help user.

Context Sensitive Help

You can double-click the QuickHelp executable or a help file to launch QuickHelp. More often, a user wants to view a help topic related to a specific feature in the application.

Once the help file has been developed and tested on the various platforms, and accounts for the slight user interface and feature differences, you're ready to add context sensitive links from the application. In Windows, users expect to press F1 when the cursor is positioned over a menu command, window, or dialog to see the related help topic. Likewise, you can support F1, or any approach of your choosing, for each platform using simple communication commands between your application and QuickHelp. These commands send the ID of a topic to QuickHelp, based on the user's context within the application.

Windows programs talk to QuickHelp using a couple of COM (Component Object Model) methods. COM is a Microsoft technology for sending data from one application to another that's supported by virtually every Windows programming language. Macintosh applications talk to QuickHelp utilizing a few different Apple Events. A Linux application would use a pipe for communication.

In each case, the application would need some code inserted to allow it to recognize that the F1 key was pressed, and ensure the appropriate topic ID was sent to QuickHelp using a COM method, AppleEvent, or Linux pipe. QuickHelp provides a description of these commands, and a snippet of code that can be customized for the language, or platform of choice.

QuickHelp also comes with a ready-to-use RealBasic QuickHelpController class, a Delphi TQuickHelpController class, and a Kylix THelpViewer class. To complete a multi-platform help authoring project, use these source code classes, or create something similar, in your own application code. Finally, add the appropriate topic IDs to each menu command, window, and dialog, then test to ensure that the appropriate topics get displayed in QuickHelp.

Conclusion

To create a good help system, a developer needs to understand what a help system is, and what it is not. The purpose of a help system is to quickly inform the user, not entertain them with flashy pictures, or moving video. Pictures can add a professional touch when used to guide and reinforce understanding, but don't waste screen space with snapshots that are quickly outdated when new features are added to your application.

Help topics should be short, cohesive, and to the point. Don't make users scroll through long topics to find the information they want, or expect them to read your help information page by page like a book. Users go to a help system when they have a specific question or problem, so most topics should be very task oriented. Listen to the questions users ask when learning to use your software, and make sure there's a topic that addresses each one, a descriptive title that clearly identifies it, and keywords that make it easy to locate.

If you're developing an application that will forever run on only one platform, you have many programming languages, IDEs, and help authoring tools to choose from. However, if your application resides in a multi-platform world, then you'll need to carefully weigh the alternatives for developing and maintaining your code and help system.

In addition to simplifying the authoring process, QuickHelp makes it easy to create and maintain a single help file that can be deployed on different operating systems, supports multiple product editions, and works for applications written in virtually any programming language. It provides a consistent, professional help environment for both users and developers.

By Ron Davis

Book Review: PHP and MySQL Web Development

It used to be you were an HTML "programmer" if you understood what <B> was. I've never really thought there was any such thing as a HTML "programmer." HTML is a mark-up language. It's data, not instructions. I'm not going to get into my personal feelings on scripting verses programming, but I've done a bit of one and a lot of the latter. The day came when my personal website needed to be able to generate content on the fly and I went looking for a language to make it do so.

At first what I wanted was C/C++ but that really isn't what you want to do web work in. Remember you are mostly manipulating text, which C/C++ isn't really all that good at. Also you are generally working with a machine you can't compile on. You can upload files though, and those files can be executed, so you need a scripting language.

Perl is the language everyone associates with UNIX, but after attempting to remove my cerebellum through my eye sockets during early Perl sessions I went looking for another language to write in. That's when I found PHP. Here was a C like scripting language made to generate web pages - one that integrated with databases, for even more interesting possibilities.

There are a lot of good online tutorials on the web for PHP. It is a very popular language. This is partially because it is free and partially because it doesn't induce suicidal tendencies when you try to code in it.

While there is no real link between PHP and MySQL they always seem to go together. This is mainly because they are both free and offered as part of Linux, the dominant platform for web hosting companies. So everyone with a web site gets PHP and MySQL for free.

When it comes time to get serious about learning to use these two technologies, you should pick up the book *PHP and MySQL Web Development* by Luke Welling and Laura Thomson. It is both a great introduction and reference for PHP.

Personally I like books that teach you by doing. I want examples that will provide real world use. Even when a book does an example a chapter, I will often change the example to something I need as I read. *PHP and MySQL Web Development* does this well. After the required introductory chapter, each chapter introduces a new concept with a relevant example. In the end you have a complete web site with lots of different functionality.

To use the book you are going to have install PHP and MySQL. If you are using a non-server version of MacOS X you will have to install MySQL. There is downloadable package to do this with instructions at Marc Liyanage's site:

http://www.entropy.ch/software/macosx/mysql/

If you want to run PHP code on your local machine using the built in Apache server, you will need to do some configuration. Apple provides some instructions on how to do this.

http://developer.apple.com/internet/macosx/php.html

Once this is done you should have no trouble doing any of the examples in the book. I found no platform-specific problems while using the book.

The first part of the book teaches you the basics of PHP, including how you put PHP code inside your HTML file. It goes on to discuss the various features of the language includeing manipulating strings with regular expressions, reading and writing files, and arrays. After you get a basic understanding of PHP, it switches to talking about MySQL.

This first section on MySQL deals only with designing and creating databases. You work on the command line with MySQL.

After you have learned to use MySQL, you'll learn how to talk to MySQL with PHP. There are two chapters in the MySQL section showing basic and advanced interaction. In reality talking to other databases is accomplished almost exactly the same way. I had a site I built on my local machine using MySQL and then moved to a server and used it with MSSQL by changing the prefix on the calling routines.

Part three of the book is the meat of learning to use these technologies. The authors step you through a complete e-comerce system built with PHP and MySQL. By the time you

Ron Davis is a long time Macintosh Software Engineer, having worked for companies like Apple and Metrowerks on a variety of products from development tools to anti-virus software. His day job is working for Alsoft, and his evening job is R.A.D. Productions, makers of Suck It Down and FinderEye.

get to this point in the book you understand the language and MySQL enough you can just start pulling the sections of this program you need out and use them. If you skip around you will occasionally have to go back to a previous chapter to figure out how the database is set up or what some variables mean. But if you need a user system, refer to chapter 24 "Building User Authorization and Personalization." Shopping cart? Chapter 25. Content management system? Chapter 27.

Appendix A on installing PHP and MySQL is largely useless to the Mac user. The index is good, letting you use the book as a general PHP/SQL reference. When you want more depth you can check out the PHP manual online at:

```
http://www.php.net/docs.php
```

Appendix B also gives an extensive list of web resources.

This book sits on my bookshelf with little post it notes sticking out at relevant pages. The only real complaint I have about the book is its size with over 800 pages. But it does cover a lot of stuff. Also since it is example based, it is somewhat hard to find a specific PHP function. I have little bookmarks for the date command and the SQL Alter table command.

If you are looking for a good book to cover all of the things about PHP and MySQL you need to do most common web tasks, this book is it.

By Kevin Hemenway, Unquenchable Excellence

Enabling CGI Scripts, The Second

You've enabled CGI, but how do you know it's good?

In the last issue, we learned about CGI scripts: what they are, what they can do, how they're already enabled within Apache, and how to tweak that configuration to be more URL friendly. What we didn't do is teach you anything for the future: at most, we brought a wide-eyed wonder-boy to a patch of poison ivy, and backed away slowly. Will he rub it on his skinned knee? Pin it to little Susie's dress as a token of his affection? Roll around in it like catnip? *Where is the inbred fear necessary for every child's survival?*

Insert transitional one-liner here!

Dissection—Similarities

Before we can understand, be aware, and watch for the security ramifications of running CGI scripts from unknown and untrusted third parties, we need to see how they're coded, how poorly written ones can ruin our mornings, and how to look for some semblance of quality. The quickest way to get a general feel is with the two sample scripts already installed with Apache: **/Library/WebServer/CGI-Executables/printenv** and **/Library/WebServer/CGI-Executables/printenv/test-cgi**. If you looked at their source code last month, you may have noticed they're written in two different languages.

The smaller of the two scripts, **test-cgi**, starts with **#!/bin/sh**, whereas **printenv** instead uses **#!/usr/bin/perl -T**. These lines, specifically the **#!** prefix, are often called the "shebang", and tell us which interpreter will execute the programming instructions that follow. The interpreter located at **/bin/sh**, rarely seen in production CGI, indicates that the rest of the code is written in the shell scripting language. Any CGI script you deploy will need to have some sort of shebang—whether it's **/bin/sh**, **/usr/bin/perl**, **/usr/bin/python** or something else entirely, it's absolutely required. Not only is it necessary, it also has to be accurate: if your only Perl is **/sw/bin/perl**, then the shebang should point there instead. Shebangs can also contain command line arguments: in **printenv**, **-T** is passed directly to the **/usr/bin/perl** interpreter (where it means something we'll cover a bit later).

Another similar difference between our two scripts is the printing of something called a *Content-type* (**Listing 1**), which tell the requesting user-agent (your visitor's browser) what sort of data it's about to receive (an image to render, text to display, XML to parse, etc.). The Content-type will never actually be shown in your final output—it's hidden pixie dust for the browser's benefit only (if you're curious, Mozilla allows you to view the Content-type by getting the "Page Info" of the current URL). Without this crucial bit of contextual magic (and the two required newlines), Apache will fail your CGI scripts with an "Internal Server Error". This error is never a satisfying explanation—you'll need to check Apache's **/var/log/httpd/error_log** for the exact reasoning.

Listing 1: Printing the Content-type in Shell and Perl

From the sample CGI scripts printenv and test-cgi

```
# content type display from test-cgi
# note that echo spits out a newline,
# 2 echo's for the 2 required newlines.
echo Content-type: text/plain
echo

# and the similar entry from printenv
print "Content-type: text/html\n\n";
```

The values of our Content-types (**text/plain** and **text/html**) didn't just appear out of thin air—they're *MIME types*, and most any file you've ever worked with has one. You can find a large listing of MIME types, based on their common file extensions, by perusing the **/etc/httpd/mime.types** file. For example, the matching MIME types for JPEG, XHTML, Quicktime, and Microsoft Word files are:

```
image/jpeg                  jpeg jpg jpe
application/xhtml+xml      xhtml xht
video/quicktime             qt mov
application/msword         doc
```

If you can't find the matching MIME type for the data you're interested in serving (either because it's not in the **mime.types**

Kevin Hemenway, coauthor of *Mac OS X Hacks* and *Spidering Hacks*, is better known as Morbus Iff, the creator of disobey.com, which bills itself as "content for the discontented." Publisher and developer of more home cooking than you could ever imagine (like the popular open-sourced aggregator AmphetaDesk, the best-kept gaming secret Gamegrene.com, the ever ignorable Nonsense Network), he's twirling his hair and trying not to cheerlead. Contact him at morbus@disobey.com.

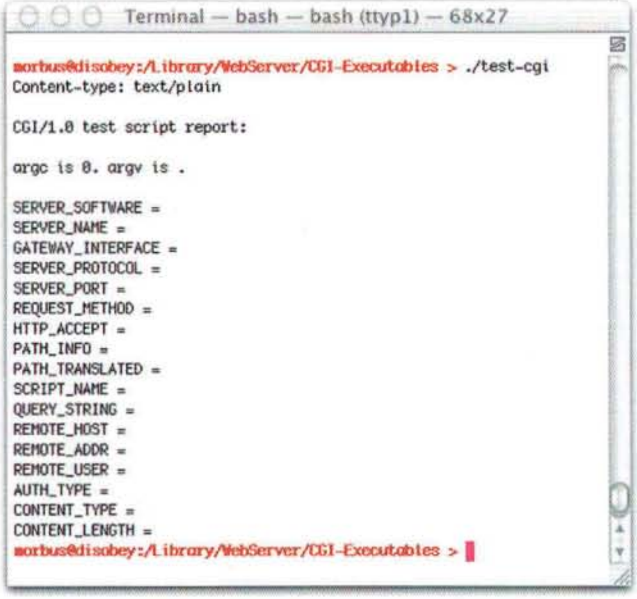
file or Google has spurned your search request), you can use the "some sort of data" MIME type of application/octet-stream. This has already been explicitly assigned to a number of files, including Apple disk images:

```
application/octet-stream    dms lha lzh exe class so dll dmg
```

Dissection—Why The Perl Script Is Arguably Stronger

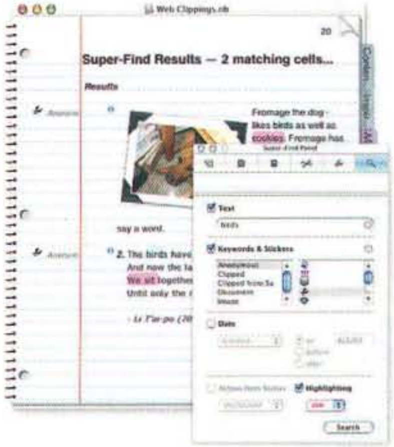
All CGI scripts, regardless of what they're programmed in, can be run from the command line—whether they actually do anything useful is a case-by-case basis. This is a surprisingly useful bit of information: since troubleshooting and debugging happens best when unfrilled by complication, removing Apache from the process can prove helpful. Running your CGI scripts on the command line can preemptively weed out problems like missing Content-type's, file permission errors, invalid syntax problems, missing language extensions, and so forth.

Both the test-cgi and printenv scripts run "successfully" at the command line, although only the first gives any useful output (**Figure 1**). Compare this to the regular browser-based output we demonstrated in the last *MacTech* (or simply re-access http://127.0.0.1/cgi-bin/test-cgi). The first line is that dastardly Content-type and, as mentioned before, is normally processed by the browser and removed from the final display. Since we're running the script without the benefit of a web server or browser, the Content-type is viewable without extra effort. This becomes a handy barometer: if you run your CGI script from the command line *and there's no Content-type*, it'll never run correctly under Apache.



```
morbus@disobey:/Library/WebServer/CGI-Executables > ./test-cgi
Content-type: text/plain

CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE =
SERVER_NAME =
GATEWAY_INTERFACE =
SERVER_PROTOCOL =
SERVER_PORT =
REQUEST_METHOD =
HTTP_ACCEPT =
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME =
QUERY_STRING =
REMOTE_HOST =
REMOTE_ADDR =
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =
morbus@disobey:/Library/WebServer/CGI-Executables >
```

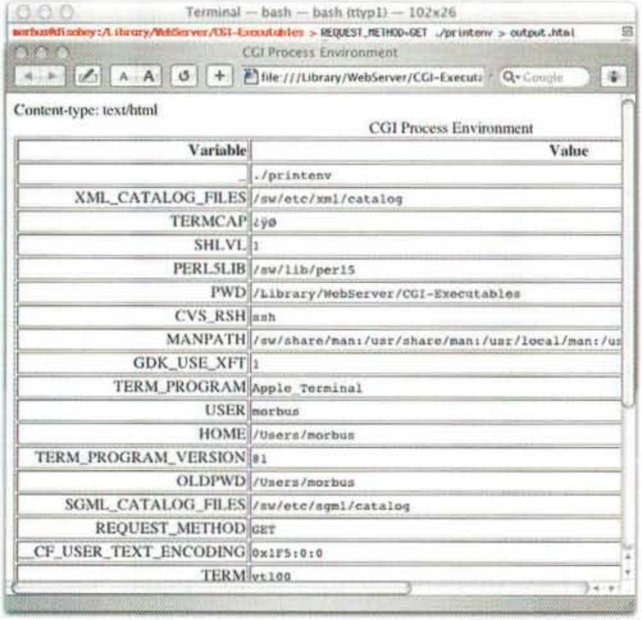
***Figure 1:** The slightly undefined test-cgi, when run in the Terminal*



But wait… there's no Content-type if we try to run **printenv** (in fact, there's nothing at all), so why does it work when we access it by URL (http://127.0.0.1/cgi-bin/printenv)? In actuality, this is one of the "strengths" of the Perl version. If you check the source code, the next line after our required shebang (ignoring comments) is:

```
exit unless ($ENV{'REQUEST_METHOD'} eq "GET");
```

This terminates the script unless it was invoked via a **GET** request. Generically speaking, unless it is a **POST**, every request a web browser makes is a **GET** with or without key/value pairs. Since the shell isn't a web browser, no **GET** is issued and the script terminates. If we wanted to get fancy, we could fake the required method by running **setenv REQUEST_METHOD GET && ./printenv** (if you're using the **tsch** shell; **REQUEST_METHOD=GET ./printenv** if you prefer **bash**). As a result, we get a Terminal full of HTML listing the environment variables. We can redirect this mass of HTML to a file by adding **> output.html** to our previous command line; **Figure 2** shows the generated file.


Terminal — bash — bash (ttyp1) — 102x26
CGI Process Environment
file:///Library/WebServer/CGI-Executa
Google

```
morbus@disobey:/Library/WebServer/CGI-Executables > REQUEST_METHOD=GET ./printenv > output.html
```

Content-type: text/html

CGI Process Environment

Variable	Value
_	./printenv
XML_CATALOG_FILES	/sw/etc/xml/catalog
TERMCAP	¿yø
SHLVL	1
PERL5LIB	/sw/lib/perl5
PWD	/Library/WebServer/CGI-Executables
CVS_RSH	ssh
MANPATH	/sw/share/man:/usr/share/man:/usr/local/man:/us
GDK_USE_XFT	1
TERM_PROGRAM	Apple_Terminal
USER	morbus
HOME	/Users/morbus
TERM_PROGRAM_VERSION	81
OLDPWD	/Users/morbus
SGML_CATALOG_FILES	/sw/etc/sgml/catalog
REQUEST_METHOD	GET
__CF_USER_TEXT_ENCODING	0x1F5:0:0
TERM	vt100

Figure 2: *Shell output of our tricked printenv script*

Figure 2 also gives us another reason why the Perl script is stronger: it doesn't pretend to know what the environment is going to look like. **test-cgi**, hard-coded to display the values of known variables (**SERVER_SOFTWARE**, **SERVER_NAME**, **GATEWAY_INTERFACE**, etc.), shows nothing but undefined values when run from the Terminal (**Figure 1**), where those specific entries don't normally exist.

Three Ways Perl CGI Scripts Can Be Improved

The bulk of the code within the **printenv** script caters to creating a pretty HTML page, something not important to the true purpose of generating a list of the current environment. To make our upcoming improvements more clearly, we'll base our changes on the Perl script shown in **Listing 2**, which does the exact same thing as **printenv**, only without the HTML. For all intents and purposes, this is a working CGI script: it's got the shebang pointing to the correct Perl interpreter, and it prints a plain-text Content-type before any other data.

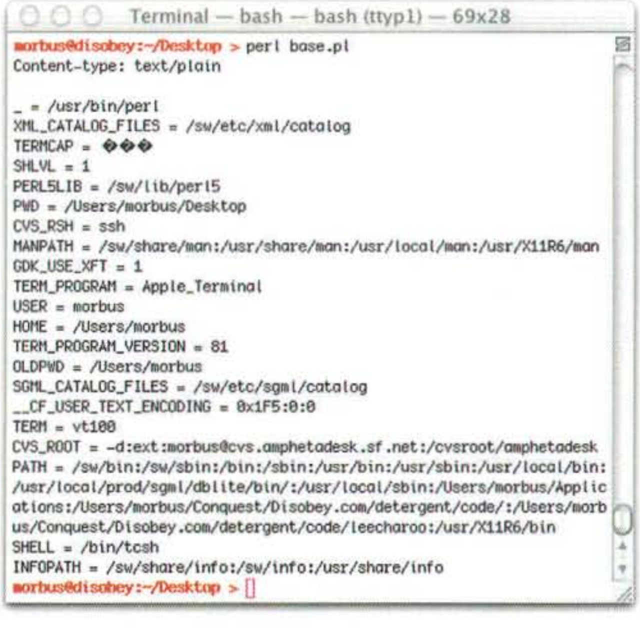
Note that even though we're talking specifically about CGI scripts, the following improvements can, and should, be made in most any Perl script, especially those to be used in production environments. Security should never be a feature.

Listing 2: Printing the environment more simply

Our base.pl script could use some improvements.

```
#!/usr/bin/perl

print "Content-type: text/plain\n\n";

foreach $var (keys %ENV) {
   print "$var = $ENV{$var}\n";
}
```

Save this file as **base.pl** and run it from the command line; my output is in **Figure 3**. None of our upcoming improvements will change this display and, as you can see by comparing it to **Figure 2**, it's identical save for the loss of HTML (and the differences between Safari and the Terminal's interpretation of **TERMCAP**).



```
morbus@disobey:~/Desktop > perl base.pl
Content-type: text/plain

_ = /usr/bin/perl
XML_CATALOG_FILES = /sw/etc/xml/catalog
TERMCAP = ���
SHLVL = 1
PERL5LIB = /sw/lib/perl5
PWD = /Users/morbus/Desktop
CVS_RSH = ssh
MANPATH = /sw/share/man:/usr/share/man:/usr/local/man:/usr/X11R6/man
GDK_USE_XFT = 1
TERM_PROGRAM = Apple_Terminal
USER = morbus
HOME = /Users/morbus
TERM_PROGRAM_VERSION = 81
OLDPWD = /Users/morbus
SGML_CATALOG_FILES = /sw/etc/sgml/catalog
__CF_USER_TEXT_ENCODING = 0x1F5:0:0
TERM = vt100
CVS_ROOT = -d:ext:morbus@cvs.amphetadesk.sf.net:/cvsroot/amphetadesk
PATH = /sw/bin:/sw/sbin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:
/usr/local/prod/sgml/dblite/bin/:/usr/local/sbin:/Users/morbus/Applic
ations:/Users/morbus/Conquest/Disobey.com/detergent/code/:/Users/morb
us/Conquest/Disobey.com/detergent/code/leecharoo:/usr/X11R6/bin
SHELL = /bin/tcsh
INFOPATH = /sw/share/info:/sw/info:/usr/share/info
morbus@disobey:~/Desktop > 
```

Figure 3: *Our rewritten script's (base.pl) output*

Our improvements to the script are quite minimal additions, but they ensure that user data has been properly checked for dangerous input, warnings have been enabled for common mistakes or typos that don't necessarily stop a script from running, and a stricter development environment has been used to encourage stronger coding and careful variable declaration. The revised script is shown in **Listing 3**.

Listing 3: Printing the environment more strongly

Our revised script is three times stronger than before.

```
#!/usr/bin/perl -wT
use strict;

print "Content-type: text/plain\n\n";

foreach my $var (keys %ENV) {
   print "$var = $ENV{$var}\n";
}
```

- **Use warnings:** The first change, adding –w to the shebang, turns on Perl's warnings pragma, which spits a list of optional, non-fatal warnings to STDERR (which becomes Apache's error_log when run as a CGI). Technically, you don't *have* to address any of the messages since the script will continue on regardless, but they'll alert you to typos, uninitialized values, deprecated functions, and a slew of other mishaps that can eventually escalate into full-blown bugs. Typically, the messages are terse enough to be useful for seasoned Perl programmers, but you can increase their verbosity by adding use diagnostics; within the body of your code.

- **Use strict:** Our third and fourth changes complement our warnings. Perl's strict pragma should be used in any script that is more than "casual", and ensures that every variable is pre-declared and localized, and that other "unsafe constructs" are detected and addressed. Unlike warnings, any error that triggers strict will stop your script from continuing further. You'll notice that we've localized our $var variable with the my() function. The first time you use strict, it'll feel like an unwieldy and overly doting mother, but scripts that compile cleanly benefit from an attention to detail that strengthens their quality immensely.

- **Use taint:** Even though it is "*strongly* recommended", very few Perl or CGI scripts use *taint mode*, which is what the –T on the shebang enables. Under this mode, any outside data received by your code is considered highly dangerous, and will cause script errors until it has been checked for safety. These safety checks can be as simple as ensuring that a command line argument only contains alphanumerics, or that the process you're spawning isn't being handed potentially damaging shell metacharacters. While taint mode will force you to focus more strongly about the evils of the outside world and exactly what data you expect, programmers who misunderstand how to "untaint" data may inadvertently do so incorrectly, creating a false sense of security.

These programming additions aren't the ultimately panacea, but merely a placebo. Yes, your code will be stronger with them, but that doesn't mean crucial bugs won't creep in and ruin your day. Serious coders and sysadmins should take a look at the following sampling of Perl and CGI security links:

- The Perl Security manpage, accessible by typing **man perlsec** in your Terminal, can also be read online at http://www.perldoc.com/perl5.6.1/pod/perlsec.html
- "Avoiding security holes when developing an application", a six part series from LinuxFocus.org: http://www.linuxfocus.org/English/November2001/article203.shtml
- SecureProgramming (http://www.secureprogramming.com/) offers a huge collection of links to over 50 articles, books, recipes to learn from and adapt, and more.
- RFP's "Perl CGI problems", which appeared in an old issue of the seminal Phrack magazine, still remains relevant: http://www.wiretrip.net/rfp/txt/phrack55.txt
- CERT's "How To Remove Meta-characters From User-Supplied Data In CGI Scripts", in both Perl and C: http://www.cert.org/tech_tips/cgi_metacharacters.html. Handy for when you're looking to untaint some data.
- The "Securing Programming for Linux and Unix HOWTO", available from http://en.tldp.org/HOWTO/Secure-Programs-HOWTO/. Similar articles like "The Hack FAQ" (http://www.nmrc.org/pub/faq/hackfaq/index.html), and the "WWW Security FAQ" (http://www.w3.org/Security/Faq/www-security-faq.html) will also prove insightful.

Choosing a CGI Script for Deployment

The above programming suggestions are fine if you're solely looking at the code quality of a potential CGI script, but there are few more areas to investigate before you can consider a program worthy of being installed on your server:

- **Check the Bugtraq archives** (http://securityfocus.com/archive/1). Anyone interested in security should be reading Bugtraq, where a large community of hackers, white hats, sysadmins, and professionals regularly post bugs, exploits, and warnings for insecure products. Occasionally, you'll also see new whitepapers concerning various aspects of security and programming. Before installing new scripts, comb the archives to see if any advisories have been posted. If so, ensure they've been fixed before using the code.
- **Googling for problems** can prove illuminating, as you'll often find common tech support problems, heaps of praise or scorn for the code or author, and occasionally, other web hosts who offer the script for their own customer base.
- **Check the dates:** When was the script last updated? Is it so long ago that no one will give a darn if you have a problem? Just because a script doesn't have any *reported* problems in Bugtraq doesn't mean that it isn't susceptible to relatively new exploits like cross-site scripting attacks (http://www.cgisecurity.com/articles/xss-faq.shtml). Code that has been updated recently has a better chance of good turnaround time for crucial fixes, updates, and support.
- **Got logfiles?** Most CGI scripts don't have any logging capability, primarily because they only do one small thing (like email forms, add one to a number, display a calendar, etc.) Some complicated scripts, however, can benefit from logging, especially those with built-in user authentication ("who is using my site?") or flaw tracking ("a bug occurred at [time], and things turned awry [like this]"). Scripts can use their own logfiles or Perl's **Sys::Syslog** module to log directly to **/var/log/system.log**.

Homework Malignments

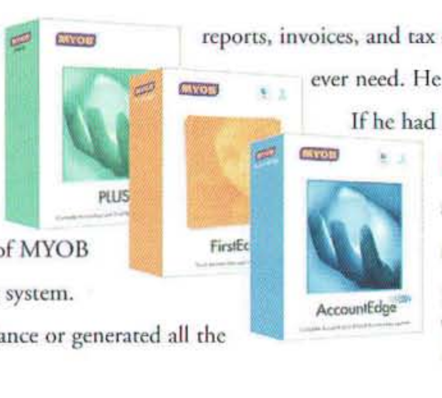
In our next column, we'll move on to configuring PHP, as well as explain the up- and downsides between forking processes (like CGI) and embedded modules (like **mod_php**). We'll explore the default configuration of PHP, the non-existent configuration file (**php.ini**) and, if we have time, how to install MySQL and do a few integration tests. For now, students may contact the teacher at morbus@disobey.com.

- Besides **-w**, you can also enable Perl's warning pragma with **use warnings**; (similar to **use strict;**). Subtle differences exist between the two—research them and find out which satisfies your programming needs better.
- Any Perl script with logging may eventually run up against a perceived "buffering" problem, the sordid details of which are explained in Mark Jason Dominus' "Suffering from Buffering?" (http://perl.plover.com/FAQs/Buffering.html).
- If you're looking to brush up on your Perl knowledge, you can't go wrong with O'Reilly's *Learning Perl*, *The Perl Cookbook* (which just received an impressive Second Edition update), and the recent *Learning Perl Objects, References, & Modules*. You can read sample chapters from all the books at http://www.oreilly.com/.

See Dick. See Dick run his business with software that wasn't written and designed for his Macintosh. Poor Dick.

A moment of silence for Dick, please. A good guy with a good small business, but his accounting software was one of those PC transcription jobs, not pure MAC like MYOB AccountEdge and MYOB FirstEdge.

If only he'd known about the amazing capacity of MYOB software to bring out the best in his MAC operating system. He could have tracked and managed finances at a glance or generated all the reports, invoices, and tax documents that he and his accountant would ever need. He could have spent more time with his clients.

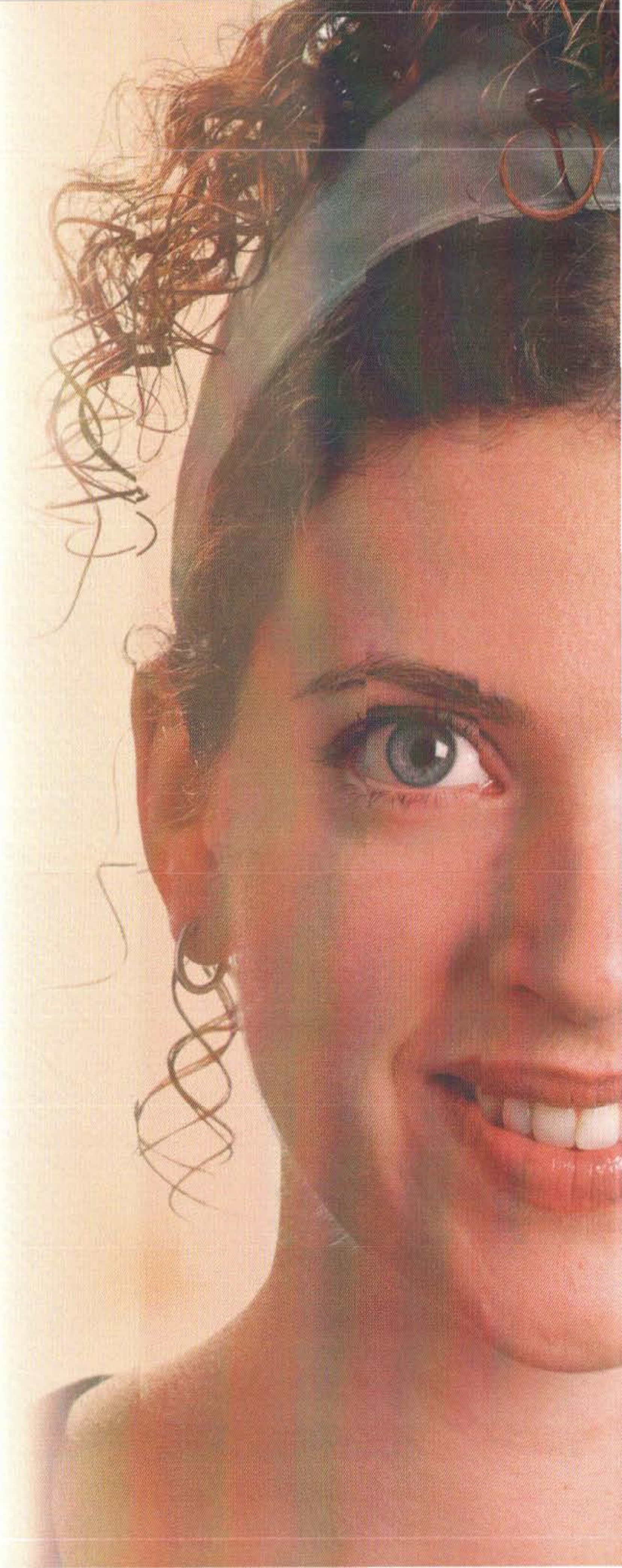


If he had only known that MYOB develops the world's best selling MAC small business management software for lots of good reasons, this story might have had a happy ending. Sorry Dick.

MYOB,
THE MAC ANSWER.

©2003 MYOB US, Inc. (800)322-MYOB www.myob.com/us

KENSINGT
Adobe
ation



By Dave Wooldridge

Building a Business

Marketing is Not a Dirty Word... Really.

Marketing and sales – the two words that make most developers cringe even more than bug reports. Why the dread? For many of us, the elusive world of marketing is unknown territory. Sure, we can spend countless hours programming complex algorithms, but hand us a marketing plan with spreadsheets and sales forecasts, and we all start to squirm in our chairs. Of course, we'd rather spend our time developing the next "killer app," but if you're an independent developer who depends on software sales to pay the bills, then marketing is an unavoidable task. With careful planning, patience, creativity, and a little help from this new monthly column, it hopefully won't seem so daunting.

The goal of this column is to demystify software marketing. Can't afford a huge advertising campaign or a dedicated sales staff? Not to worry, you don't necessarily need a big budget to build a successful software business. Each month, We'll focus on a different topic such as promotional opportunities, distribution, publicity, press releases, sales incentives, web site strategies, etc. We'll explain how to quickly and easily integrate these cost-effective concepts into your own development plans.

MYTH: Build It and They will Come

Okay, this isn't "Field of Dreams," people. So you've built an amazing shareware application. You post a download link on your web site and submit a listing to VersionTracker.com. You receive several positive comments and even net six or seven sales within the first few days. The initial flurry of activity is promising. You become optimistic, thinking this could be your ticket. This could be your flagship product that helps you launch the software empire you've always dreamed of running. Another week goes by and sales dwindle into nothingness. No more downloads. No e-mails. It's as if your product doesn't even exist. It's lost in the endless sea of the VersionTracker archives – its only chance of surfacing again is when you announce your next update. Only if there's no demand, why put the energy into updating a dead product? And then you remember that other great shareware idea you had and decide to pursue that instead. Sound familiar?

As developers who pour our blood, sweat, and tears into programming, we often have little energy left to focus on marketing the final product. After slaving away for six months to finish one application, there's an overwhelming excitement to unleash your new baby to the world, eager to receive affirmation from users that all the hard work was worth it. But in the rush, important steps that can help ensure a long and successful life for your product are often bypassed or overlooked.

No developers want to see their products fail. Even though we all realize that marketing is crucial, the dilemma is often how and where to start. Looking for a good book on the subject? Good luck! Most business books tend to describe abstract marketing concepts that cater to the widest spectrum of readers, leaving software developers uncertain as to how these general themes can be applied to their specific marketing needs. To assist our readers, MacTech will offer a different approach. This column will show you practical, cost-effective marketing techniques and opportunities throughout every stage of a fictional software product's life cycle, illustrating real-world examples.

While this first installment may seem elementary to some veteran developers, it should serve as a good entry point for those programmers new to the software business before we dive into specific topics in the following months.

Do Your Homework

Marketing does not start with the final product. It begins with the initial concept. Ever think of a cool software idea, only to find out that twelve similar applications have already been released? Sure, many of them are probably amateurish attempts compared to your product, but the problem is that they exist nonetheless. The saturated market increases consumer confusion – sometimes there are just too many choices. Obviously, your competitors are going to boast that

Dave Wooldridge is the founder of Electric Butterfly (www.ebutterfly.com), the web design and software company responsible for Stimulus, HelpLogic, UniHelp, and the popular developer site, RBGarage.com.

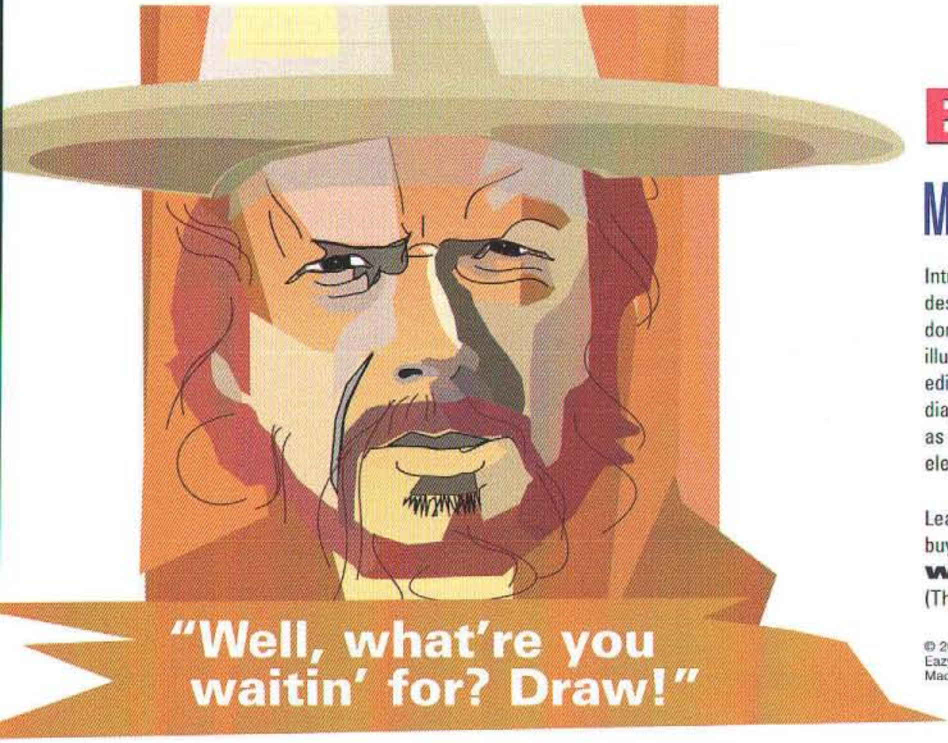
their products are the best, so how many different software applications will consumers be willing to evaluate before stumbling upon yours? And will they even get to that point? Or will they give up somewhere along their quest and pick some mediocre product that meets only some of their needs, not knowing your product will better serve them? People are typically impatient and desire quick solutions when shopping for software. If they can't easily find the diamond among the rocks, they will often settle for anything that remotely glitters.

The point here is that if you're going to release a product in an already over-saturated market, then you need to devise a way to make your product stand out from the pack. There should be a unique benefit or feature that sets your product apart from its competitors. Let's look at the crowded field of software note keepers. These handy little utilities allow you to store and quickly access notes, passwords, clipboard items, images, sounds, and more. Search for "notes" on VersionTracker.com and more than 50 product listings will appear. Most of them claim to be the most time-saving, general purpose notes organizer on the market, so how will your application be different? What will be the key-selling point that makes your product special?

To answer those questions, we must start with research. Investigate the software category you're planning on entering. Now it goes without saying that the best niches to fill are those that remain untapped. It's fairly easy to dominate a specific market if you have no competitors, but let's assume you're entering the crowded note keeper category, which already has several key leaders and dozens of fledging underdogs. Before you even think of opening your IDE, before writing a single line of code, your first job is to thoroughly research each and every one of your competitors.

Visit your competitors' web sites and read any reviews and related online forums you can find. Figure out what it is about the market leaders that help them outsell the other products in that field. Is it a specific feature set? A polished, easy-to-use interface? An unbeatable price? A combination of elements? The game is about filling a niche or need that isn't being properly addressed. Your goal is to convince as many people as possible that your software is not only a worthy choice, but the ONLY choice for them to consider.

Even though we're developers, we're also consumers. If you were to purchase a software utility for organizing small chunks of data, what would *you* use it for? Storing code snippets, maybe? Ah, we might be onto something here. Yes, you could easily paste important code fragments in any notes keeper, but undoubtedly, some vital features would be missing. It would be nice if these code snippets could be viewed with syntax coloring. And it would be nice if the interface was small and compact enough to run alongside your screen-hogging IDE, giving you quick access to your library of



code snippets. Sounds like a great product for software developers and web site designers.

Search VersionTracker for "code snippets" and this time, only two results are found. Not only have we defined attractive key-selling points tailored for a specific target market, but we also found a niche category with very few competitors.

Before jumping into development, we should stop to consider our commitment to this particular project. Growing a loyal customer base for a new product takes a lot of patience and hard work. As the developer, you have to determine if this project will sustain your interest over years of dedicated programming and labor poured into continuous updates, customer support, and marketing. Do you have creative ideas on what new features to add in version 2.0? Any thoughts on version 3.0? It is important to develop a long-term game plan for the product's future releases. This will not only give you a development direction so that you can properly build an extensible application framework to accommodate those future plans, but it will also help you gauge your interest level.

So who cares if the product lasts three years or three weeks? Your customers care! This is where your dedication to continue supporting a product directly affects your marketing and sales success. If you stop product development and support after only the first 1.0 release due to a lack of interest, your abandoned customers will become skeptical and untrusting. They may never buy another of your products – a fate you should desperately try to avoid since the bigger goal here is to build a profitable software company, not drive business away.

A new product should never be dropped after only one release because of disappointing sales. Instant success rarely happens right out of the gate with version 1.0. It takes time to spread awareness for your new product, so your dedication to the project must remain strong. Just remember that maintaining the happiness of your existing customers is very important since you will be depending on them for upgrade sales and spreading positive comments about your software to their friends, co-workers, etc.

So now that we've committed our time and resources to this project, the next step is to give our fictional code snippet organizer a name. A good name is extremely important and can make or break a product's success. Put a lot of thought into it. Write down at least a dozen different names. Why so many? Because I can guarantee that most of them will already be taken. Perform searches for each one of your name ideas (with and without spaces) on VersionTracker.com, MacUpdate.com, Google, and the U.S. Patent & Trademark Office (www.uspto.gov). If a name is already in use, then scratch it from your list, even if it's not a software product. Companies are very protective of their registered names and trademarks (as they should be). You don't want to spend six months of development time and your hard-earned marketing money, only to receive a "cease and desist" letter from someone's lawyer.

The name should help describe the function of the product, so that if a consumer sees the name listed in a catalog or web site, it might pique their interest even if a description is not available. With this in mind, it's probably important to include the word "code" since we are attempting to appeal to programmers. "Code" might be best coupled with another word that represents a collection, a library, or some sort of storage device. CodeBase, CodeBank, CodeFarm... all taken. After perusing the dictionary and thesaurus, I find the word "quiver." Hmm, CodeQuiver. Has a nice ring to it and it rolls off the tongue quite easily. Plus, the symbolism is perfect for the application. A quiver gives an archer quick access to arrows (your code) when firing at a bullseye target (your programming task).

Brand Identity

Now that we've established the development direction and a name, our next assignment is to design a product logo and icon. The objective is to tie a specific image to the application that becomes synonymous with the product. This visual cue should be unique and simple in nature, something that can be easily identified by consumers after repeatedly seeing it in advertisements and online.

Why is visual branding so important? How many times have you said, "I can't remember that person's name, but I would definitely recognize their face." We all tend to rely quite heavily on visual stimulus, so why not benefit from such a simple marketing tool? If the goal is for consumers to remember your software name, then provide them with a visual cue to help solidify it in memory.

A perfect example of a successful branding strategy is the popular FTP program, Fetch. Most Mac users are so familiar with the little brown dog with the blue disk in his mouth that Fetch Softworks' current advertisements (which appear regularly in MacTech) do not even mention Fetch's core functionality. One ad simply shows the Fetch mascot with the title "Good Doggie." Although Fetch has continuously evolved along with the latest Mac OS advancements, its brand identity has always remained consistent (see **Figure 1**). The visual connection reinforces consumer recognition. Over the years, Mac users have gradually grown to identify the Fetch name and icon with FTP.

Establishing a new brand does not happen overnight. You may release several versions before your software's logo and icon become familiar to potential buyers. It's about frequency and continuous exposure. That's why getting reviewed in magazines and getting listed on software sites is so important for shareware titles. Even though Fetch Softworks' current marketing strategy includes advertising, the branding of Fetch's little brown dog was being emblazoned into the minds of Mac users long before the recent print ads. Back in the early days of the Mac, Fetch was one of the first, easy-to-use FTP applications. It was originally developed by Jim Matthews and released as shareware from Dartmouth College (where he worked). With only word of mouth, Fetch became one of the defacto FTP utilities for Mac users. Every release featured the little brown dog with the blue disk in its mouth. When Jim Matthews purchased the Fetch source code and name from Dartmouth College in early 2001 and started Fetch Softworks, the Fetch mascot was already one of the most recognized icons in the Mac community.

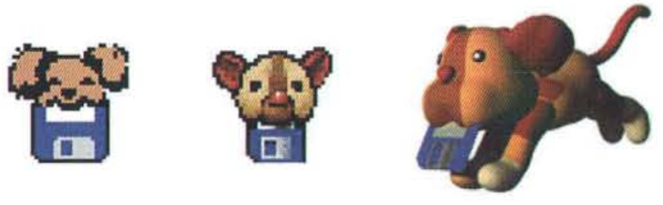


Figure 1. *From the early days of Mac (Left icon) to Mac OS 9 (Middle icon) to Mac OS X (Right icon), the well-known brand identity of the popular FTP program, Fetch, has remained consistent.*

So how does one establish a visual brand? Include your product logo and icon in everything! Don't stop at your application icon. Integrate the imagery into your software's interface, documentation, advertisements, your web site, etc. Everywhere your product name is mentioned, its related icon should accompany it whenever possible. Obviously, things like text-only press releases are the exception to this rule.

In designing an icon for our fictional CodeQuiver product, the name makes the task quite easy. An archery target with an arrow in the bullseye provides a powerful visual that helps reinforce the product name, as well as a simple image that will still be easy to identify when shrunken down to a small 32 x 32 icon (see **Figure 2**). Even though Mac OS X utilizes beautiful 128 x 128 pixel icons, there are times when the icon will be viewed in the Finder in sizes as small as 16 x 16 pixels. An effective icon is one that looks good at any standard icon size and resolution. Remember, unidentifiable icons do not add branding power to your product.

Since the icon imagery will supply most of the colorful eye-candy, the title treatment should remain clean and simple so that the overall logo does not appear cluttered and busy. Using a sans-serif font, only the "Q" is enlarged, with its bottom "hook" straightened to the same angle as the arrow to make the "Q" look like a quasi bullseye target (see **Figure 2**).

Don't have a single artistic bone in your body? Not to worry, there are lots of freelance artists out there that specialize in logo and icon design. With the Internet bringing the world to your fingertips, it's fairly easy to find an inexpensive graphic designer online. Many shareware developers have benefited from their own customers who have offered to redesign their interface icons in exchange for a free license. And if your product sells for less than $100, then that's the cheapest design work you'll ever find! While this is a great trade for you, it's unfortunately a bad indication that your product is being ill received. If the top feature request is an interface redesign, then acquiring help from a professional icon designer – even if you have to pay for it – will be worth every penny in the long run.

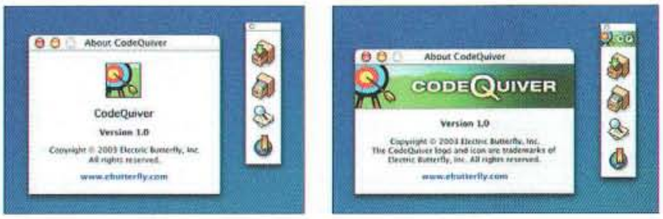
Your Interface is a Marketing Tool

Whenever the words "interface" and "marketing" are used in the same sentence, there are usually a handful of programmers who cry foul, afraid that the imaginary church and state line between development and business is being blurred. If that results in selling more software licenses, then how can this be a bad thing?

If you're a shareware developer, then many of your users may stumble upon your product on a shareware compilation CD-ROM or a software archive site (I'll discuss how to get your software titles distributed to all of those vital outlets in a future column). Without the benefit of seeing your web site first, the application download serves as the user's first exposure to your product. You can add a flaming red icon to the included "Read Me First" file, but odds are, users will lazily bypass any documentation and simply double-click on your application, knowing nothing about your product except its name and a brief one-line description.

So it essentially comes down to first impressions. A user's first impression of your interface is going to be one of the primary factors in deciding whether or not to purchase a license. When dealing with shareware and trial versions that consumers can freely evaluate, your application's interface becomes the most important marketing tool in your arsenal. The interface is it's own best advertisement, so take the time to ensure that it looks good, while reinforcing your product's brand identity.

To cater to our target market, we've designed CodeQuiver's interface as a compact, vertical palette that can sit in the corner of the desktop screen alongside your favorite IDE. We could closely follow Apple's Human Interface Guidelines and develop a fairly generic Aqua interface (Left example in **Figure 2**) or we could interject some branding to give the interface a little personality (Right example in **Figure 2**). More cries of blasphemy arise from those developers who firmly clutch the Human Interface Guidelines as if it were their bible of truth. But even Apple has defied its own guidelines from time to time. Nothing's written in stone. That's why they're called "guidelines" and not "rules."



About CodeQuiver

CodeQuiver

Version 1.0

Copyright © 2003 Electric Butterfly, Inc.
All rights reserved.

www.ebutterfly.com

About CodeQuiver

CODEQUIVER

Version 1.0

Copyright © 2003 Electric Butterfly, Inc.
The CodeQuiver logo and icon are trademarks of
Electric Butterfly, Inc. All rights reserved.

www.ebutterfly.com

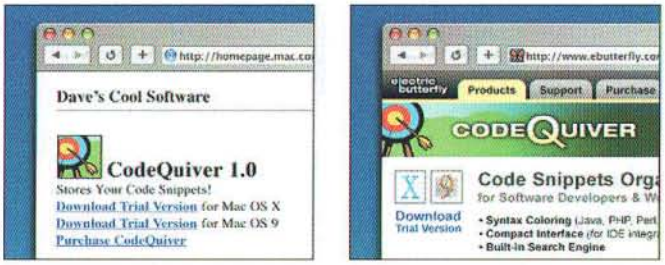
Figure 2. *Sometimes breaking the rules is good business. Although the Left example may conform to Apple's Human Interface Guidelines, the Right example provides a professional polish that enhances the user experience.*

Notice how all of the best-selling Mac software titles have their own unique interfaces? They all look like standard Mac applications, but each and every one of them defies at least one item in Apple's Human Interface Guidelines. Their unique look and feel is what gives them personality. Consumers may not immediately notice the subtle niceties of a specific interface, but it does subconsciously affect their user experience. The integrated branding – whether it's illustrated through unique toolbar icons or through the creative placement of the product logo/icon – adds a professional polish to your application that may enhance an otherwise generic interface (Right example in **Figure 2**).

Through the Looking Glass

While we're on the subject of first impressions, the brand identity of your product should also be properly represented on your web site. If people follow an URL from a magazine review or advertisement to your site, only to find a very basic web page with little more than a download link and a version number, you're doing a major disservice to your software business.

Take a look at the two web page examples in **Figure 3**. The Left example provides minimal product information with only a one-line description. Interested users are then expected to download and evaluate the software. Most people are extremely busy and don't want to waste the time downloading and testing an application just to learn more about it (especially if they use a slow dial-up modem). At this point, many of these people lose their initial interest and turn away... and you've just lost those potential sales! And why? Because your web site did not provide the detailed product information that consumers were looking for.


http://homepage.mac.co

Dave's Cool Software

CodeQuiver 1.0

Stores Your Code Snippets!

Download Trial Version for Mac OS X

Download Trial Version for Mac OS 9

Purchase CodeQuiver

http://www.ebutterfly.co

Code Snippets Org

for Software Developers & W

Download Trial Version

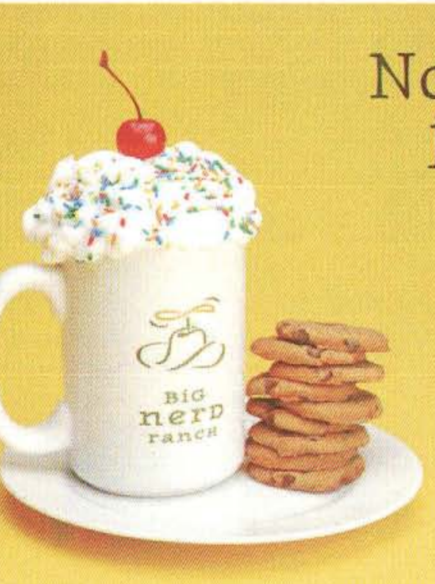
- Syntax Coloring (Java, PHP, Perl
- Compact Interface (for IDE integr
- Built-in Search Engine

Figure 3. *Which site would you trust more with your credit card? Consumers see your web site as a looking glass into how you run your business.*

When you shop online for software, what are the key pieces of information you want to see? At a minimum, this should include a brief product description with a detailed list of features, a few screenshots of the application in action, system requirements, price, and what kind of customer support is offered. The Right example in **Figure 3** shows an enhanced web page design that includes these key pieces of information. The purpose of your web site is to not only provide a download location for your software, but to help promote and sell your products. Your job as a marketer and salesperson does not stop until a customer purchases your software (and even then, you'll want to keep that sales channel open for future upgrade offers). This means that everything that leads a customer to that purchase point must continue to influence their decision, and your web site is a major part of that equation.

Your web site is your storefront to the world. People judge your software and your business by the way your web site looks and functions. If your site appears too basic and sloppy (as in **Figure 3**'s Left example), it gives the perception of being an amateur shareware outfit. If the developer does not care enough to spend time and money on his/her business, then why should a consumer? Having an elegantly designed web site will not only make your business and products appear more professional, but it will instill something very valuable in your customers: TRUST. Again, it's always about first impressions. When comparing the two examples in **Figure 3**, which site would you trust more with your credit card? Which site looks like a business that will continue to support their products one year from now? Two years from now?

If you don't have much experience creating web graphics and HTML, then it's in the best interest of your software business to either hire the assistance of a freelance web designer or learn how to do it yourself. Macromedia Dreamweaver and Adobe GoLive are just a few of the great



web site creation and management tools that can make the task easier.

Still using that free community web space from .Mac, AOL, Earthlink, GeoCities or Tripod? Most consumers are savvy enough to spot the telling URL address of one of these free hosting accounts (especially if you list it in an advertisement). It is yet another element that oozes the word "amateur" or "hobbyist." If you want people to perceive you as a professional software company, then registering a domain name (such as www.mycompany.com) is a wise investment. At only $35 a year (average US cost), domain names should be an essential part of your marketing mix. You can register product-related domain names like www.codequiver.com, but your marketing efforts should always refer to your company domain name since you want people to associate your products with your company. This is important when launching a new product in the hopes that customers will remember how much they like your other software and assume the new CodeQuiver must be just as cool.

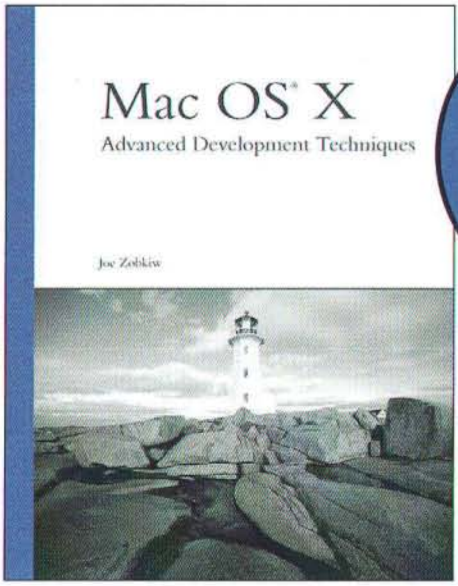
Some developers believe that fancy web sites are a waste of time because in the end, it's the quality of the software itself that ultimately convinces users to buy a license. While this is true, it does not account for 100% of the decision making process. Most people know who Macromedia and Adobe are, but few shareware companies have that kind of name recognition. Consumers fear the unknown, so as an unknown entity, your job is to alleviate that fear through your marketing messages and your web site. Customers are not just buying a product, they are investing in your business, trusting that you will continue to support and update your product.

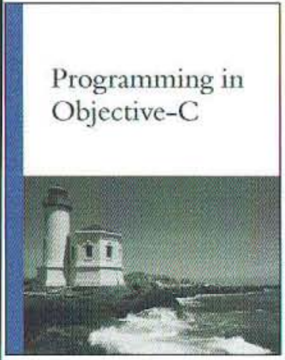
Think Big Picture

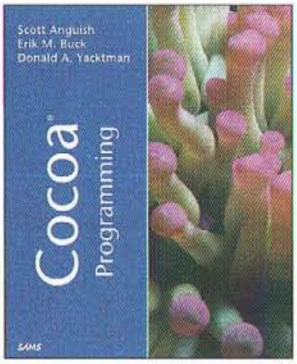
Before diving into specific topics in future issues, the goal of this first installment was to illustrate the importance for independent developers to see the "big picture." Building a successful software business requires a lot of strategic planning. While research and interface design are typically considered part of the product development process, everything you do up until the initial release directly affects the marketing and lifespan of your product. Establishing a brand identity and positive first impressions for your software and business should never be afterthoughts. Think about your marketing strategy throughout every phase of product development. Having a well-conceived game plan will save you from the many development headaches that impulsive decisions usually cause. If you always think "big picture," you're one step closer to a long-lasting product line.

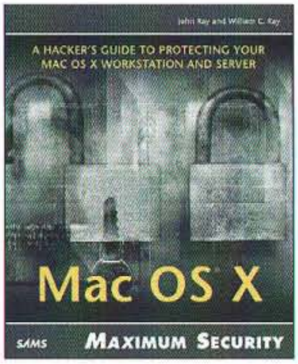
Next month, we'll explore several marketing tools and tricks that you can integrate into your web site to help increase awareness and sales for your software.

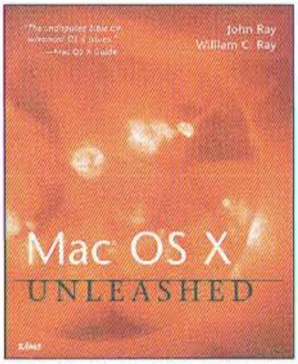
The Fetch icons appear courtesy of Jim Matthews, Fetch Softworks.


Mac OS X
Advanced Development Techniques
Joe Zobkiw


Programming in Objective-C


Scott Anguish
Erik M. Buck
Donald A. Yacktman
Cocoa
Programming
SAMS


John Ray and William C. Ray
A HACKER'S GUIDE TO PROTECTING YOUR MAC OS X WORKSTATION AND SERVER
Mac OS X
SAMS
MAXIMUM SECURITY


John Ray
William C. Ray
Mac OS X
UNLEASHED
SAMS

QUICKTIME
TOOLKIT

by Tim Monroe

Children of the Revolution

Editing QuickTime Movies with Revolution

Introduction

In the previous *QuickTime Toolkit* article ("Revolution" in *MacTech*, September 2003), we took a first look at Revolution, a rapid application development tool published by Runtime Revolution Ltd. We saw how to create a new application — which we called RunRevVeez — that can open and display QuickTime movies. We saw how to set things up so that the user can have several movies open at once, and we saw how to use a few of the built-in Revolution commands to modify the appearance of a movie player object at runtime. In terms of movie *playback*, RunRevVeez is just about complete.

The situation with movie *editing* is somewhat different, however. As I mentioned last time, Revolution has no built-in support for editing QuickTime movies. In addition (as far as I can tell), it provides no support for tracking changes to a window or document, and it provides no way to save an edited movie. We'd certainly like our application to be able to handle these tasks, so we'll have to go beyond the built-in capabilities of Revolution. We need to write a Revolution *plug-in*.

Happily, Runtime Revolution provides a software development kit (SDK) for writing Revolution plug-ins, and this makes writing our plug-in a snap. With just a few dozen lines of new C code and a handful of routines borrowed from our existing C-based sample application QTShell, we'll be able to handle all the basic editing operations, keep track of the modification state of a movie window, and save edited movies into new files.

Unhappily, even with this plug-in, there are a few things we won't be able to accomplish with Revolution. The Revolution runtime engine opens QuickTime movie files with read-only permission, which effectively prevents us from saving any changes to a movie into the file we opened the movie from. We will be able to write an edited movie into a new file. (In a nutshell, we'll be able to implement the "Save As" menu item but not the Save menu item.) Also, the Revolution runtime engine installs a movie controller action filter procedure, which effectively prevents us from installing our own procedure. This restricts our ability to access many important QuickTime capabilities. (You may recall that REALbasic currently has this same limitation; see "Basic Instinct" in *MacTech*, February, 2003.)

In this article, we'll continue our development of RunRevVeez. We'll implement the editing operations on a movie, which requires us to develop a plug-in and then to call the plug-in from within our scripts. We'll look at the file-handling operations (principally, "Save As" and Close) in the next article.

One final note before we begin: Runtime Revolution has recently released Revolution version 2.1. In these articles, I've used version 2.0.2. I would assume that the plug-in and Revolution project will work unchanged under 2.1, but I have not actually verified that.

Revolution Plug-Ins

The Revolution runtime engine is based largely on an existing product called MetaCard, which was introduced in 1990 as a competitor to Apple's HyperCard. Not surprisingly, the plug-in architecture used by MetaCard, and hence Revolution, is identical to that introduced by HyperCard. HyperCard can be extended by adding modules of commands and functions called *externals*. A set of external commands is called an *XCMD* and a set of external functions is called an *XFCN*.

Originally, XCMDs and XFCNs were packaged as executable code resources that were added to the resource fork of the application or to the resource fork of a stack. MetaCard and Revolution followed this example through Revolution version 1.1.1. In version 2.0 and later, the packaging of externals was changed; in current versions, externals on Mac OS X are packaged as *bundles*, which can be copied into the application bundle.

The packaging actually doesn't really matter all that much, since it will be taken care of by the project files provided with the plug-in SDK. The current SDK provides project files for both CodeWarrior and Project Builder. In this article, we'll work with the Project Builder version, whose project window is shown in **Figure 1.** (Notice that I've renamed the project as "QTExternal".) We'll need to modify only one file here, external.c. The file XCmdGlue.c contains a number of support routines for the external; we won't need to call any of those routines.

Tim Monroe is a member of the QuickTime engineering team at Apple. You can contact him at monroe@mactech.com. The views expressed here are not necessarily shared by his employer.

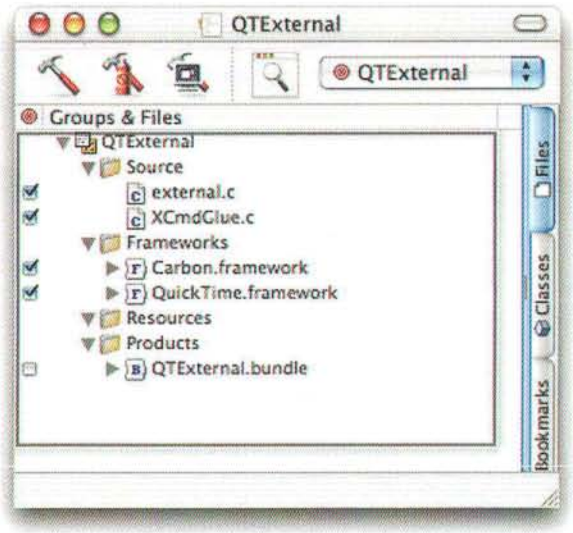


***Figure 1:** The Project Builder project*

Connecting to the Runtime Engine

Our Revolution external will define a number of procedures and functions that can be called by RunRevVeez scripts. To expose those routines to the runtime engine, we need to declare two global variables, Xname and Xtable. The Xname variable specifies the name of the external:

```
char Xname[] = "QuickTime Revolution External";
```

The Xtable variable contains an array of procedure specifiers. Each entry in the array specifies information about a single external function or command. Here's our array:

```
Xternal Xtable[] = {
   {"mcInitialize", XCOMMAND, 0, XCMD_MCInitialize,
                                               XCMD_Abort},
   {"mcUndo", XFUNCTION, 0, XCMD_MCUndo, XCMD_Abort},
   {"mcCut", XFUNCTION, 0, XCMD_MCCut, XCMD_Abort},
   {"mcCopy", XCOMMAND, 0, XCMD_MCCopy, XCMD_Abort},
   {"mcPaste", XFUNCTION, 0, XCMD_MCPaste, XCMD_Abort},
   {"mcClear", XFUNCTION, 0, XCMD_MCClear, XCMD_Abort},
   {"selectAll", XCOMMAND, 0, XCMD_SelectAll, XCMD_Abort},
   {"selectNone", XCOMMAND, 0, XCMD_SelectNone,
                                               XCMD_Abort},

   {"mcEnableEditMenuItem", XFUNCTION, 0,
                  XCMD_MCEnableEditMenuItem, XCMD_Abort},

   {"windowSetModified", XFUNCTION, 0,
                  XCMD_SetWindowModified, XCMD_Abort},
   {"saveAs", XFUNCTION, 0, XCMD_SaveAs, XCMD_Abort},
   {"", XNONE, 0, NULL, NULL}
};
```

The first item in a procedure specifier is the name of the routine that we'll use in our scripts. The second item indicates the type of routine; it's XCOMMAND for commands (which do not return a value to the caller) and XFUNCTION for functions (which do return a value to the caller). The third entry is used by the runtime engine and should be set to 0 by our external. The fourth entry is the name of the corresponding C language routine in the external. (In other words, it's the routine that is called when our script executes the first item.) Finally, the fifth item is the name of an *abort routine*, which is called when the user cancels the execution of an external routine. All our external routines will use the same abort routine, shown in **Listing 1.**

Listing 1: Handling user cancellations

XCMD_Abort

```
void XCMD_Abort()
{
  DebugStr("\pQuickTime Revolution External abort");
}
```

Our abort routine just prints a diagnostic message on the standard error output.

Handling Commands

When a script calls the mcInitialize command (for instance), the external function XCMD_MCInitialize is executed; XCMD_MCInitialize has this declaration:

```
void XCMD_MCInitialize (char *args[], int nargs,
        char **retstring, Bool *pass, Bool *error);
```

The first parameter passed to XCMD_MCInitialize is an array of C strings that specifies the parameters that were passed to the mcInitialize command. The second parameter specifies the number of items in that array. We'll call mcInitialize with only one parameter, like this:

```
put the movieControllerID of player "MoviePlayer" \
          of stack newStackName into mc
mcInitialize(mc)
```

The third parameter, retstring, is a pointer to a C string that contains the results of the external routine. For procedures, this is ignored by the runtime engine; for functions, this string is returned to the script as the command result. The buffer for this string must be allocated by the external and is disposed of by the runtime engine.

The fourth and fifth parameters are used to pass other information back to the runtime engine. The pass parameter indicates whether we want the command (in this case, mcInitialize) to be passed up the message hierarchy after it is executed. In general, we shall return false in this parameter. The error parameter indicates the success or failure of the external routine. Once again, we'll always pass back false, to indicate that no error occurred. (Errors may indeed occur within our external routines, but RunRevVeez will have no capability to work around errors; so there's little point in letting it know that something went wrong.)

Configuring the Movie Controller

So let's see how we can implement the handler for the mcInitialize command. As we've seen, the args parameter will contain a single C string, which is the movie controller identifier encoded as a string. To get a value of type MovieController, we need to convert the string to a long.

```
mc = (MovieController)atol(args[0]);
```

Once we've got the movie controller identifier, we can call any QuickTime APIs that operate on a movie controller. In RunRevVeez, we need to enable editing (by calling MCEnableEditing) and enable keyboard event handling (by calling MCDoAction with the mcActionSetKeysEnabled selector). **Listing 2** shows our complete handler for the mcInitialize command.

Listing 2: Initializing the movie controller

XCMD_MCInitialize

```
void XCMD_MCInitialize (char *args[], int nargs,
        char **retstring, Bool *pass, Bool *error)
{
  MovieController mc = NULL;
  ComponentResult result = noErr;
  char *retstr = NULL;

  // initialize the movie controller as desired
  *pass = false;
  *error = false;
  if (nargs == 1) {
    mc = (MovieController)atol(args[0]);

    if (mc != NULL) {
      // enable editing
      result = MCEnableEditing(mc, true);

      // enable keyboard event handling
      MCDoAction(mc, mcActionSetKeysEnabled, (void *)true);

      // disable drag support
      MCDoAction(mc, mcActionSetDragEnabled,
                                            (void *)false);
    }
  }

  retstr = calloc(1, 2);
  if (retstr != NULL)
    retstr[0] = (result == noErr) ? '0': '1';

  *retstring = retstr;
}
```

As indicated just above, we set both pass and error to false. And we pass back, via retstring, a C string of length 1 that contains either "0" or "1". RunRevVeez ignores that value.

Once we've successfully called mcInitialize, the thumb in the controller bar will change to reflect that editing is enabled (as seen in **Figure 2**).

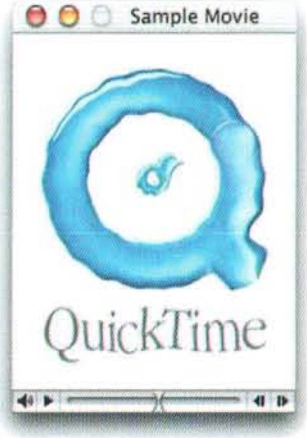


Figure 2: *A movie window with editing enabled*

Handling Edit Operations

So, we've enabled movie controller editing. Now we need to handle the various editing operations. In these cases, we need to pass a value back to the caller, indicating whether the operation completed successfully. That's so RunRevVeez can know to set the movie window as modified and that the movie has changed since last opened or saved. We'll return the string "1" if the edit operation fails and "0" if it succeeds. **Listing 3** shows how we'll handle the mcUndo command.

Listing 3: Undoing a movie edit

XCMD_MCUndo

```
void XCMD_MCUndo (char *args[], int nargs,
         char **retstring, Bool *pass, Bool *error)
{
  MovieController mc = NULL;
  ComponentResult result = noErr;
  char *retstr = NULL;

  *pass = false;
  *error = false;
  if (nargs == 1) {
    mc = (MovieController)atol(args[0]);

    if (mc != NULL)
      result = MCUndo(mc);
  }

  retstr = calloc(1, 2);
  if (retstr != NULL)
    retstr[0] = (result == noErr) ? '0': '1';

  *retstring = retstr;
}
```

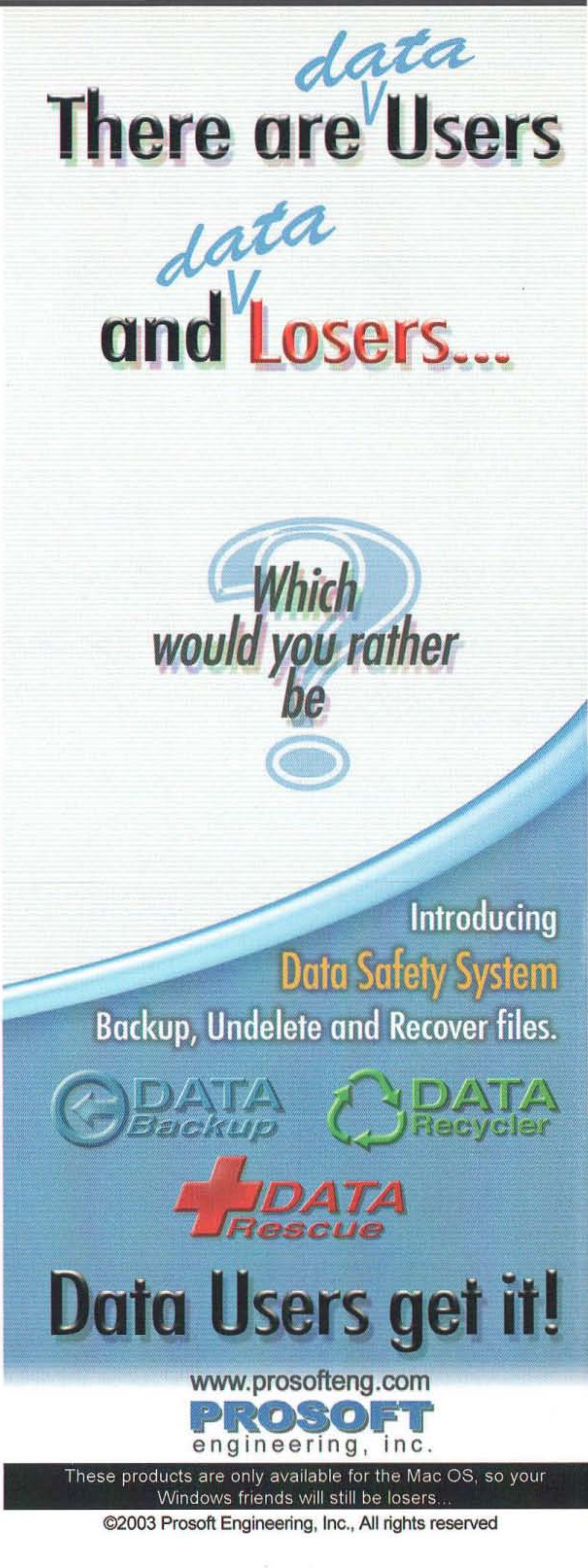
We simply retrieve the movie controller identifier and call MCUndo. Then we call calloc to allocate a 2-byte buffer, to hold the returned character and the null terminating byte.

The other editing operations are quite similar. **Listing 4** shows how we handle the mcCut command, and **Listing 5** shows how we handle the mcCopy command. Notice in both cases that we call PutMovieOnScrap to place the cut or copied movie segment onto the scrap.

Listing 4: Cutting a movie selection

XCMD_MCCut

```
void XCMD_MCCut (char *args[], int nargs,
         char **retstring, Bool *pass, Bool *error)
{
  MovieController mc = NULL;
  ComponentResult result = noErr;
  Movie editmovie = NULL;
  char *retstr = NULL;

  *pass = false;
  *error = false;
  if (nargs == 1) {
    mc = (MovieController)atol(args[0]);

    if (mc != NULL) {
      editmovie = MCCut(mc);
      result = (editmovie != NULL) ? result: invalidMovie;
    }
  }

  // place the cut movie segment onto the scrap
  if (editmovie != NULL) {
    PutMovieOnScrap(editmovie, 0L);
    DisposeMovie(editmovie);
  }
```



```
  retstr = calloc(1, 2);
  if (retstr != NULL)
    retstr[0] = (result == noErr) ? '0': '1';

  *retstring = retstr;
}
```

Listing 5: Copying a movie selection

XCMD_MCCopy

```
void XCMD_MCCopy (char *args[], int nargs,
        char **retstring, Bool *pass, Bool *error)
{
  MovieController mc = NULL;
  ComponentResult result = noErr;
  Movie editmovie = NULL;
  char *retstr = NULL;

  *pass = false;
  *error = false;
  if (nargs == 1) {
    mc = (MovieController)atol(args[0]);

    if (mc != NULL) {
      editmovie = MCCopy(mc);
      result = (editmovie != NULL) ? result: invalidMovie;
    }
  }

  // place the copied movie segment onto the scrap
  if (editmovie != NULL) {
    PutMovieOnScrap(editmovie, 0L);
    DisposeMovie(editmovie);
  }

  retstr = calloc(1, 2);
  if (retstr != NULL)
    retstr[0] = (result == noErr) ? '0': '1';

  *retstring = retstr;
}
```

Implementation of XCMD_MCPaste and XCMD_MCClear is left as an easy exercise for the reader. (The complete code for the QuickTime external is of course contained in the source code accompanying this article.)

Selecting All or None of a Movie

Our Edit menu contains two further items, "Select All" and "Select None", which are once again easy to implement. In earlier articles, we've seen how to handle these items by calling MCDoAction with the mcActionSetSelectionDuration selector. **Listing 6** shows how our Revolution external handles the selectAll command, and **Listing 7** shows how our Revolution external handles the selectNone command.

Listing 6: Selecting all of a movie

XCMD_SelectAll

```
void XCMD_SelectAll (char *args[], int nargs,
        char **retstring, Bool *pass, Bool *error)
{
  MovieController mc = NULL;
  Movie mv = NULL;
  ComponentResult result = noErr;
  TimeRecord tr;
  char *retstr = NULL;

  *pass = false;
  *error = false;
  if (nargs == 1) {
    mc = (MovieController)atol(args[0]);

    if (mc != NULL) {
      mv = MCGetMovie(mc);
      if (mv) {
        tr.value.hi = 0;
        tr.value.lo = 0;
        tr.base = 0;
        tr.scale = GetMovieTimeScale(mv);
        result = MCDoAction(mc,
                    mcActionSetSelectionBegin, &tr);

        tr.value.hi = 0;
        tr.value.lo = GetMovieDuration(mv);
        tr.base = 0;
        tr.scale = GetMovieTimeScale(mv);
        result = MCDoAction(mc,
                    mcActionSetSelectionDuration, &tr);
      }
    }
  }

  retstr = calloc(1, 2);
  if (retstr != NULL)
    retstr[0] = (result == noErr) ? '0': '1';

  *retstring = retstr;
}
```

Listing 7: Selecting none of a movie

XCMD_SelectNone

```
void XCMD_SelectNone (char *args[], int nargs,
        char **retstring, Bool *pass, Bool *error)
{
  MovieController mc = NULL;
  Movie mv = NULL;
  ComponentResult result = noErr;
  TimeRecord tr;
  char *retstr = NULL;

  *pass = false;
  *error = false;
  if (nargs == 1) {
    mc = (MovieController)atol(args[0]);

    if (mc != NULL) {
      mv = MCGetMovie(mc);
      if (mv) {
        tr.value.hi = 0;
        tr.value.lo = 0;
        tr.base = 0;
        tr.scale = GetMovieTimeScale(mv);
        result = MCDoAction(mc,
                    mcActionSetSelectionDuration, &tr);
      }
    }
  }

  retstr = calloc(1, 2);
  if (retstr != NULL)
    retstr[0] = (result == noErr) ? '0': '1';

  *retstring = retstr;
}
```

Enabling and Disabling Edit Menu Items

RunRevVeez needs to enable and disable the Edit menu items according to the edit state of the movie in a movie window. For instance, when a movie is first opened and no edit operations have yet occurred, the Undo item should be disabled. QuickTime provides the MCGetControllerInfo function, which we've used in the past to adjust the states of our edit menu items. We'll use it again here, as shown in **Listing 8.**

Listing 8: Adjusting the Edit menu

XCMD_MCEnableEditMenuItem

```
void XCMD_MCEnableEditMenuItem (char *args[], int nargs,
        char **retstring, Bool *pass, Bool *error)
{
  MovieController mc = NULL;
  ComponentResult result = noErr;
  long mcInfo = 0L;
  short index = 0;
  char *retstr = NULL;

  *pass = false;
  *error = false;
  retstr = malloc(2);     // either "0" or "1", plus the terminating null byte

  if (nargs == 2) {
    mc = (MovieController)atol(args[0]);
    index = (short)atoi(args[1]);

    if (mc != NULL)
      result = MCGetControllerInfo(mc, &mcInfo);
  }

  switch (index) {
    case kUndoItemIndex:
      retstr[0] = mcInfo & mcInfoUndoAvailable ? '1': '0';
      break;

    case kCutItemIndex:
      retstr[0] = mcInfo & mcInfoCutAvailable ? '1': '0';
      break;

    case kCopyItemIndex:
      retstr[0] = mcInfo & mcInfoCopyAvailable ? '1': '0';
      break;

    case kPasteItemIndex:
      retstr[0] = mcInfo & mcInfoPasteAvailable ? '1': '0';
      break;

    case kClearItemIndex:
      retstr[0] = mcInfo & mcInfoClearAvailable ? '1': '0';
      break;

    case kSelectAllItemIndex:
    case kSelectNoneItemIndex:
      retstr[0] = mcInfo & mcInfoEditingEnabled ? '1': '0';
      break;

    default:
      DebugStr("\pGOT AN INDEX WE DIDN'T EXPECT!");
      break;
  }

  // tack on the terminating null byte
  retstr[1] = 0;

  *retstring = retstr;
}
```

Notice that our code here looks for *two* parameters, which are the movie controller identifier and the index of the menu item we want information about. If, according to MCGetControllerInfo, the menu item with that index should be enabled, XCMD_MCEnableEditMenuItem passes back the string "1"; otherwise it passes back the string "0".

In RunRevVeez, the code that enables or disables the menu items is contained in the script attached to the menu item group (and not to any particular menu or item). That's because, when the user clicks on the menu bar, a mouseDown message is sent to the menu item group. We want to call mcEnableEditMenuItem for each menu item index and adjust the menu item according to the value returned by it.

Listing 9: Adjusting the Edit menu

mouseDown

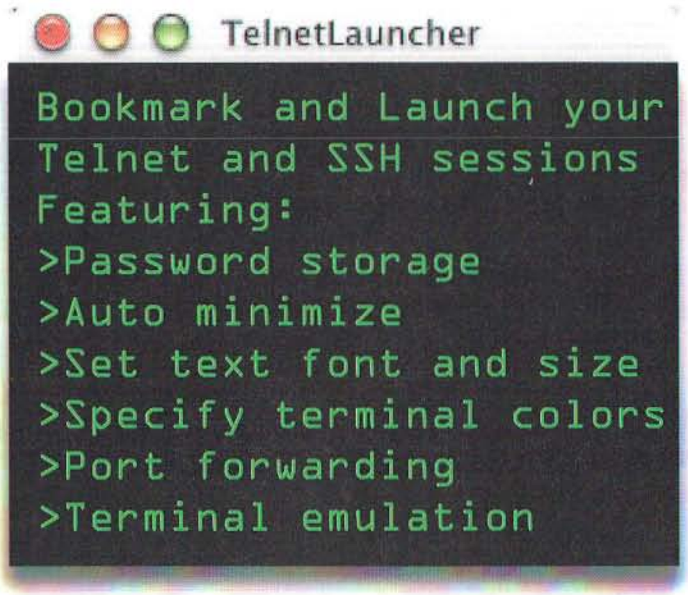
```
on mouseDown
  put first line of the openStacks into theTopStack
  put exists(player "MoviePlayer" of stack theTopStack) \
           into gotPlayer

  repeat for each item itemIndex in "1,3,4,5,6,8,9"
    if gotPlayer then
      if mcEnableEditMenuItem(the movieControllerID of \
            player "MoviePlayer" of stack theTopStack, \
              itemIndex) is "1" then
        enable menuItem itemIndex of menu "Edit"
      else
        disable menuItem itemIndex of menu "Edit"
      end if
    else
      disable menuItem itemIndex of menu "Edit"
    end if
  end repeat

end mouseDown
```

We also need to adjust the states of the items in the File menu and the Movie menu. We'll postpone our consideration of the File menu to the next article. We can handle the Movie menu as shown in **Listing 10.**

Listing 10: Adjusting the Movie menu

mouseDown

```
if gotPlayer then
  enable menuItem kShowBarItemIndex of menu "Movie"
  enable menuItem kHideSpeakerItemIndex of menu "Movie"
else
  disable menuItem kShowBarItemIndex of menu "Movie"
```



```
  disable menuItem kHideSpeakerItemIndex of menu "Movie"
end if
```

Here we use a few constants that we've defined in our message handler:

```
constant kShowBarItemIndex = 1
constant kHideSpeakerItemIndex = 2
```

Setting the Window Status

In the Aqua interface, a window's close button contains a dot if the movie in the window has been modified since opened or last saved (compare **Figure 3** with **Figure 2**).

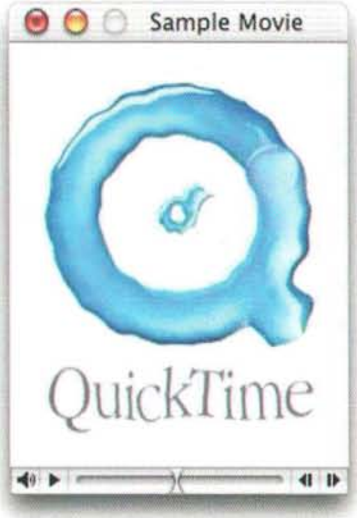


Figure 3: *A modified movie window*

In earlier *QuickTime Toolkit* articles, we've seen that we can set the window modification state by calling SetWindowModified. With Revolution, we need to call into our external to do this. **Listing 11** shows our definition of the XCMD_SetWindowModified function.

Listing 11: Setting the window modification state

XCMD_SetWindowModified

```
void XCMD_SetWindowModified (char *args[], int nargs,
        char **retstring, Bool *pass, Bool *error)
{
  WindowPtr wID = NULL;
  Boolean state;
  OSErr result = noErr;
  char *retstr = NULL;

  *pass = false;
  *error = false;
  if (nargs == 2) {
    wID = (WindowPtr)atol(args[0]);
    state = (Boolean)atoi(args[1]);

    if (wID != NULL)
      result = SetWindowModified(wID, state);
  }

  retstr = calloc(1, 2);
  if (retstr != NULL)
    retstr[0] = (result == noErr) ? '0': '1';

  *retstring = retstr;
}
```

This is pretty much like all the external procedures we've seen so far, except that the first parameter here is of type WindowPtr. Our call to windowSetModified looks like this:

```
get windowSetModified(windowID of stack theTopStack, 1)
```

A stack's windowID property contains the operating system ID of the window containing the stack; on Mac OS, this ID is a window pointer. (By the way, notice that we invoke the windowSetModified command by passing it as an expression to the get command. The "get *expr*" command is a shortcut for the expression:

```
put expr into it
```

We need to treat windowSetModified as a function, since that's how we declared it. If we had declared it as a command, we would omit the get.)

We also need to keep track of a window's modification state within our scripts in RunRevVeez (so, for instance, we know whether to enable or disable some of the items in the File menu). We could implement yet another function in our external that calls IsWindowModified. Or we can define a *custom property* associated with the movie window stack that keeps track of this modification state. Let's use a custom property. Open the movie window's property inspector palette and select the "Custom Properties" panel in the pop-up menu. The original panel looks like **Figure 4.**

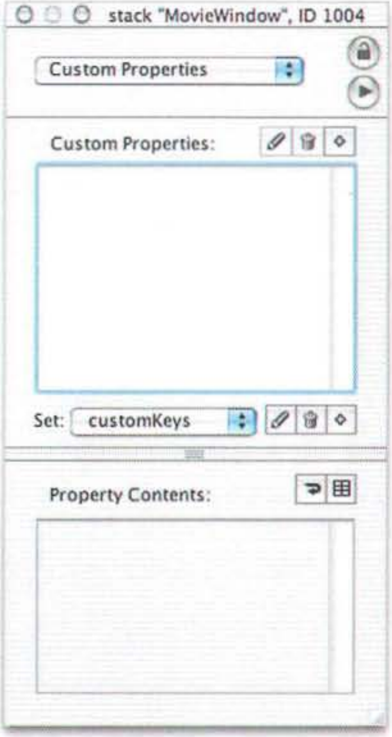


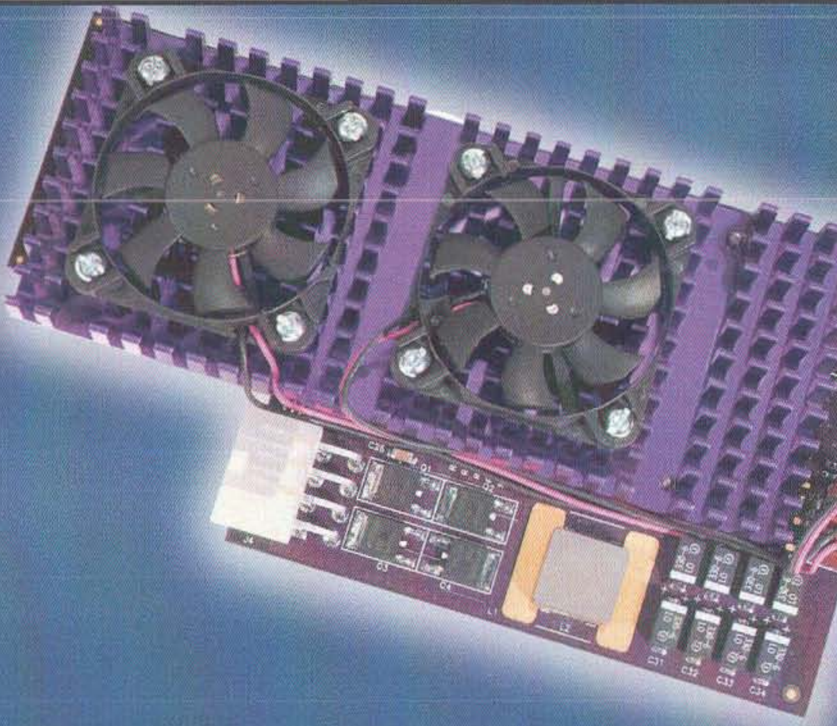
Figure 4: *The movie window's custom properties (original)*

SONNET™

SIMPLY FAST™

Upgrade Cards

*Put a G4 in your iMac,
Beige, Blue and White,
or other Mac.*



Yea ... it feels like that.

www.devdepot.com/upgrades

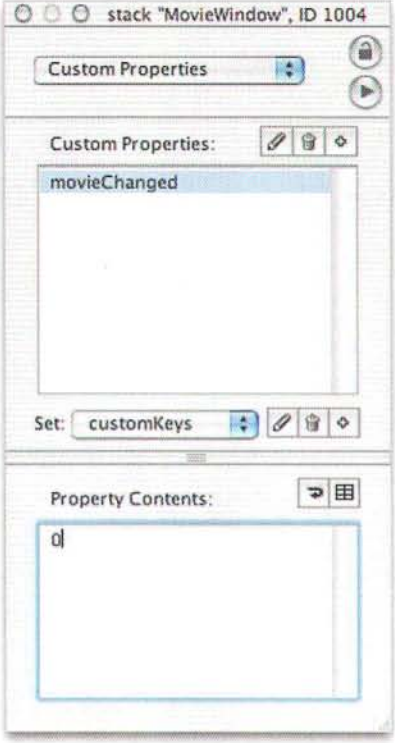
DEV DEPOT®

www.devdepot.com

877-DEPOT-NOW

Toll Free: 877-337-6866
Outside US/Canada: 805-494-9797
Fax: 805-494-9798 • orders@devdepot.com
PO Box 5200 • Westlake Village, CA 91359-5200

Click the "+" icon to add a new property. Let's call the new property movieChanged. When a movie window is first opened, this property should be set to 0, so set the property contents accordingly. The property inspector palette now looks like **Figure 5.**



stack "MovieWindow", ID 1004

Custom Properties

Custom Properties:

movieChanged

Set: customKeys

Property Contents:

0

Figure 5: *The movie window's custom properties (final)*

Once we've done this, we can access the movieChanged property just like we access any of the built-in properties, for example like this:

```
set the movieChanged of stack theTopStack to true
```

We'll see some examples of this in the next section.

Movie Editing

We're now finished constructing the movie editing portions of our Revolution plug-in module. It's very easy to put them to work. When the user selects an item in the Edit menu, the menuPick message handler of the Edit menu is called. **Listing 12** shows our complete menuPick handler. Notice that we check to make sure that the value returned by the editing operations (for example, mcCut) is the string "0", which indicates that the operation completed successfully.

Listing 12: Handling the Edit menu items

menuPick

```
on menuPick pWhich
  put first line of the openStacks into theTopStack
  if exists(player "MoviePlayer" of stack theTopStack) then
    put the movieControllerID of player "MoviePlayer" of \
                         stack theTopStack into mc
    put false into changed

    switch pWhich
    case "Undo"
      if mcUndo(mc) = "0" then put true into changed
      break
    case "Cut"
      if mcCut(mc) = "0" then put true into changed
      break
    case "Copy"
      mcCopy(mc)
      break
    case "Paste"
      if mcPaste(mc) = "0" then put true into changed
      break
    case "Clear"
      if mcClear(mc) = "0" then put true into changed
      break
    case "Select All"
      selectAll(mc)
      break
    case "Select None"
      selectNone(mc)
      break
    end switch

    if changed then
      set the movieChanged of stack theTopStack to true
      get windowSetModified \
                          (windowID of stack theTopStack, 1)
      sizeStackToMovie the short name of stack theTopStack
    end if

  end if
end menuPick
```

We also call the sizeStackToMovie method if the movie has been edited, since the size of the movie may have changed.

Conclusion

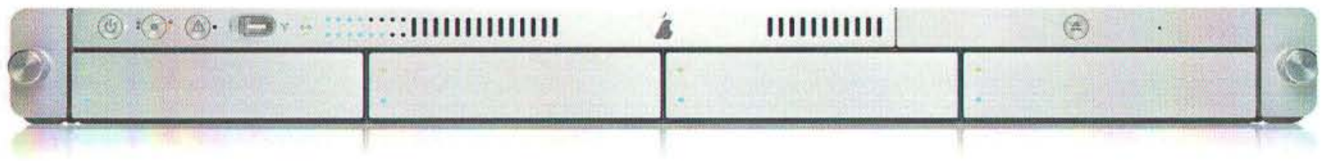
In this article, we've focused mainly on adding the ability to edit movies to our application RunRevVeez. We've seen how to construct a plug-in that allows our Revolution scripts to invoke external code modules. This is the primary avenue by which we can enhance the built-in behaviors and capabilities of Revolution.

We've got a little bit more work to do to get RunRevVeez to operate precisely as desired. We still need to handle the "Save As" and Close menu items in the File menu, and we need to tie up a few remaining loose ends. We'll tackle all that in the next article.

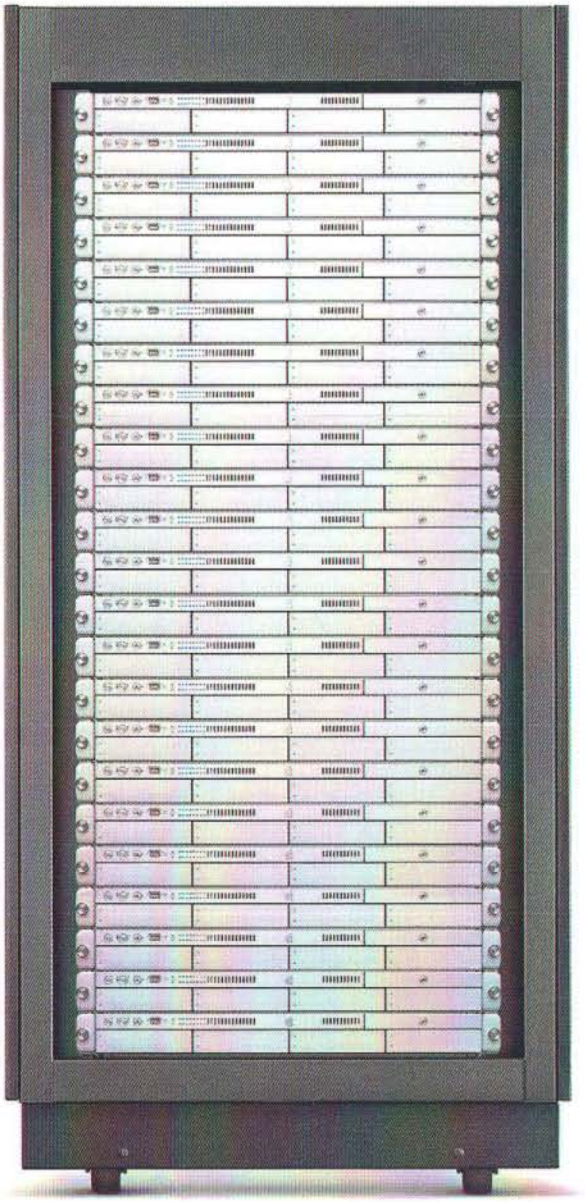
Credits

Thanks are due once again to Kevin Miller and Tuviah Snyder at Runtime Revolution Ltd. Tuviah was especially helpful with the plug-in. And a special thanks is again due to Geoff Canyon of Inspired Logic, LLC.

Xserve



Density optimized rack mounted Mac OS X Server



UNIX-based Server Solutions from Apple

Nearly half a terabyte of storage per 1U
630 gigaflops of processing power
Hot-Swappable drives
Industry-standard 1U rack-optimized design

There has never been a better time to buy...

Small Dog Electronics carries factory refurbished models too offered at substantial savings!

(subject to availability)

Small Dog Electronics
www.smalldog.com

1673 Main Street, Route 100, Waitsfield, Vermont

Apple Specialist

To learn more: http://www.smalldog.com/xserve/

By Michael R. Harvey

Laptop Cases

So, you've got the laptop of the year, now what?

Last January at MacWorld Expo San Francisco, Steve Jobs proclaimed 2003 as the year of the notebook for Apple Computer. It was no idle boast, either. Apple has some of the best portable computers on the market today. Both the 12 and 17 inch models were well received at MacWorld, and the new 15 inch Aluminum PowerBook G4 just released at Apple Expo Paris completes the line. But what do you carry it in? They are some of coolest, and most powerful laptops out there, but you still need to be able to get it, and its accessories from place to place.

I decided to look at this from a real world prospective. I developed an inventory of items any user might need and/or carry with their PowerBook on a regular basis. The list was as follows:

14" iBook
iBook power adapter
Spare iBook battery
PDA
Cell phone
iPod
Charging brick for iPod
One each Ethernet, FireWire, USB, phone, and HotSync cable
Small optical mouse
Two pens
Two CD jewel cases
Business cards
One book (I happened to be reading Bicentennial Man by Isaac Asimov at the time of this review)
Three press kits (to simulate miscellaneous paperwork)

With this pile of stuff, I set out to test some current offerings from various manufacturers to see if they could handle the load, and how well they could do it.

Brenthaven

The offerings from Brenthaven are top notch. So much so that they were the first official provider of cases to the Apple Store. These are high quality products. Well built, sturdy, and loaded with nooks and crannies for stuffing things into. Our inventory was easily handled by the Professional 17 Shoulder Case I tested. In fact, I

doubled the load (except for computer), and this case was able to carry it. The bag is very well built. Tough, ballistic nylon, with solid, easily operated zippers throughout. There is a strap for sliding the handle of your wheeled suitcase through. The case for holding the laptop is detachable from the main case. There is an accordion file for paperwork within the main compartment, several sleeves for accessories, and hardware, a pocket for a PDA, a key chain hook. and even a secret small outside pocket just big enough to hold a plane ticket, and a passport. It's got a dual handle set up for carrying, as well as a decent shoulder strap. I did not particularly care for the dual handle set up. I couldn't just grab and go. I had to collect the two parts before hauling the case away. The strap is attached to the case in such a way as to automatically carry it over your hip, a much more comfortable position. The pad on the strap is only okay. It's soft, but sticky feeling. Make sure it is resting on your shoulder exactly the way you want it because it will not slide at all. I think they meant it that way, but I found it to be a detractor. Overall, this is a truly outstanding bag, ready to handle nearly any need a user might have. It is available from the Apple Store for $149.

Kensington

These guys make all sorts of great peripherals for the Mac. The also happen to have a really nice case called the Saddlebag Pro. This bag was really a surprise. Initially, I thought it would be okay, but nothing special. I could not have been more wrong. The Saddlebag Pro is well designed, tough, durable, and can easily handle nearly anything you need it to. It has one main compartment with a padded divider for the laptop (which is able to handle up to the 17" PowerBook). The other side is spacious and easily handled many of the items from the inventory list. There is an outer area that is covered by the flap with zippers on either side. This area has places for business cards, a key chain hook, and a pocket for whatever you want. The flap itself has

pockets on either side. The main flap that covers it all also has a zippered pocket on it. The carrying handle it attached here. It is a single piece and very comfortable to grip, even when the bag is loaded down. The shoulder strap on this case was outstanding. The pad was very comfortable. Now, what about all the cable clutter? It fits in a hidden drawer that slides out of the lower side of the bag. It fit every random little piece of hardware from the list easily. Two other additions that really caught my eye. One, the bag can expand. Unzip the flap around the back edge of the Saddlebag Pro, and you give yourself even more room for those times when you have to take an entire desk worth of paperwork home with you. Last, this thing converts to a backpack. A compartment just above the suitcase handle strap opens to reveal shoulder straps that click to rings, and viola, backpack time. A nice addition for those times when you need to carry different, although I wouldn't recommend using the bag primarily in this configuration. The backpack straps are none too comfortable. The Saddlebag Pro can be found many places with an average price of around $60.

Willow Design

The Eastern Large Display Carry Case is one of many fine bags from Willow Design. All of their offerings, much like everything I tested, are made of high quality materials, and excellent workmanship. The Eastern case we tried was able to hold the inventory, but only just. It was a tight fit (some of their other larger cases would have had no problem). There are three outer pockets. One very thin one along the back of the case for holding no more than a magazine. The other two are on the front side, one atop the other. The outer most is just a zippered compartment. The larger one has pockets for anything you want to put in (the largest of which can hold a CD jewel case). There is also a zippered mesh pocket. The main compartment is where the uniqueness of this case shows. You access it by opening up the top flap that unzips on three sides (and also has another zippered mesh pocket on the underside). It is one space that is split by a solid board. The laptop rests on it, and can be secured with a Velcro strap. Flip up the board, and space for all your accessories is exposed. It's actually a nice setup. You can work on the computer as it sits in the case quite easily, accessing any papers from the pouches on the flap. The space underneath can hold most all your accessories, as long as they aren't too bulky and fit within the one of the three spaces the area is divided into. One final note on this case, the shoulder strap and single piece handle are okay, but not the most comfortable of those tested. Prices vary depending the model you choose, but the Eastern Large Display Carry Case is $90 direct from Willow Design.

Crumpler

Crumpler automatically wins in the category of cool, or insane, looking bags (depending on your take on such things). Big, bright colors, and unique names are hallmarks of the company. The test model they sent, for example, was called The Very Busy Man. This bag is all about big spaces. Everything is covered by a large flap which both Velcros and clips closed. Four pockets make up the interior of the case. From front to back: a zippered pocket that has several smaller pouches within it, that either zipper or Velcro closed, for small items. Next a large space for stuff, books, tomes, small cars. Third pocket is a flap covered space for your laptop. It, as well as the entire bag, is very well padded. This pouch can easily hold the largest PowerBook Apple ahs to offer, while also safely keeping the smallest. The last pocket is a smaller zippered pouch probably best used for holding cables, and other accessories. Back to the outside of this monster, we find only a heavy duty shoulder strap. No handle, which I found to be a big minus. The strap itself, however, is of good quality, with a sliding pad (the only one tested that has this). There is also a smaller strap from the bottom corner of the bag that hooks around the shoulder strap to help hold the case more securely to your body when you are wearing it for extended periods of time. Crumpler bags are sold mostly through camera stores. To find a dealer near you, or an online dealer, check out their web site at www.crumplerusa.com.

Does size really matter?

Bigger isn't necessarily better, at least as far as laptop cases go. It may be that you don't need to carry the Rosetta Stones, and a Volkswagen around in your bag. If you only need to get your smaller laptop, and maybe a few other things at most from here to there, then one of these could be the bag for you.

First up is McBain's Baby from Crumpler. It's one of their smaller laptop bags. It can hold a 12" iBook or PowerBook in its main compartment, and a few small bits in its zippered front pouch. The flap Is held down by Velcro. The shoulder strap is adequate, and it even has a small handle, unlike its big, and very busy, brother.

Our next item up for review in the smaller is better category is the Courtney Slipcover from Willow Design. This is a very nice bag. Like the Crumpler above, the Courtney can hold a 12" iBook or PowerBook. It has two additional compartments, however. A small accordion file with a Velcro closing flap is on the back of the case. The front space has pouches and mesh zippered compartments for accessories. There is even a detachable key ring hidden along one side of the case. Very clever.

Brenthaven had two offerings I tried out in this category. The first is the Professional 12 Shoulder Case. A very fine bag, small in size, but with a lot of places to store stuff. It can handle a 12" Apple laptop snugly, and has more places to store things than you think it should, considering its small form factor, with pouches for a PDA, and other accessories, a key hook, and even a luggage handle strap. It also has handles and shoulder strap identical to its bigger sibling. The second bag from Brenthaven was the Mobility One Messenger Bag. Much like the Professional 12, this bag has more places for stuff than you would think the laws of physics would allow. In fact, although this wasn't part of the review criteria, this bag was able to hold the test inventory I used on the larger bags. It was pretty stuffed, but it did it, and everything was nicely accessible. It is also able to hold the larger 14" iBooks in its very well padded case.

Final Analysis

Choosing a laptop case is a very personal choice. It has to fit your needs exactly, and even being off by just a little will nag at you. For the smaller bags, I went with the Willow Design bag. It was small, but still held a lot of stuff. As for the larger cases, the surprising winner for me was the Kensington Saddlebag Pro. As I mentioned earlier, I thought this bag would be adequate, but nothing special. I was totally surprised by how well this bag met my requirements. It fit all my needs exactly. Let that be a warning, when you go looking for a case, discount nothing ahead of time, because you may just eliminate the perfect case without ever knowing it.

These are just my personal preferences, though. The Brenthaven bags are outstanding. They are very likely an outstanding choice for many users. The Willow Design cases would be great for those who need smaller sizes, and need to carry a bit less than our criteria set up. Then there is Crumpler. Wacky, cool, unique, huge. Sound like your thing?

Again, it's a personal choice, but choosing from any of the cases reviewed here will definitely put a high quality, durable bag in your hands. Just make sure it fits your needs, and habits, and you will be very happy with your choice. Good luck.

MacPlay

By Michael R. Harvey

Offerings from MacPlay

Waste time, have fun

MacPlay are the folks who bring some of the greatest games you can lose hours of your life with to the Mac platform. Following is a round up of some of their current offerings.

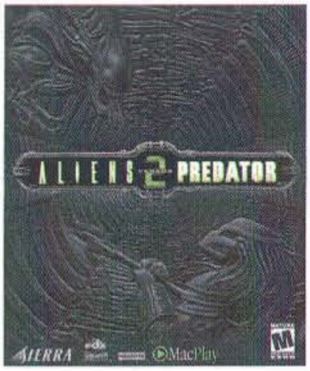
LIVE FOREVER?

No, no one lives forever, not even in video games. Particularly not in this first person shooter. In No One Lives Forever, you take on the Role of Kate Archer, femme fatale, secret agent. She exists in the world of the super spy of the 60's. This game takes its cues from all the great spy flicks of that time. This FPS is much like many others in the genre, except it's a lot more fun. Playing as a woman is different from most any other similar game, it's got a good story, and a better soundtrack. In fact, you get a bonus audio disc with some very groovy tunes on it. Well worth your time. Also note that by the time you read this, MacPlay will have released part two in this series, subtitled A Spy in H.A.R.M.'s Way, and promising to be a worthy successor to this first episode.

NOT IF THIS GUY HAS ANYTHING TO DO WITH IT

Another FPS that takes on some of the more traditional aspects of this kind of game is Solider of Fortune II: Double Helix. Here you play, and kill, as mercenary John Mullins, sent out on various missions with the ultimate goal of stopping a terrorist organization from releasing the deadly virus they have in hand. Even though it's much like many other shooters, it is a fun ride. The system requirements are fairly modest, and the graphics are quite good. If this kind of game is your thing, Soldier of Fortune II worth adding to your collection

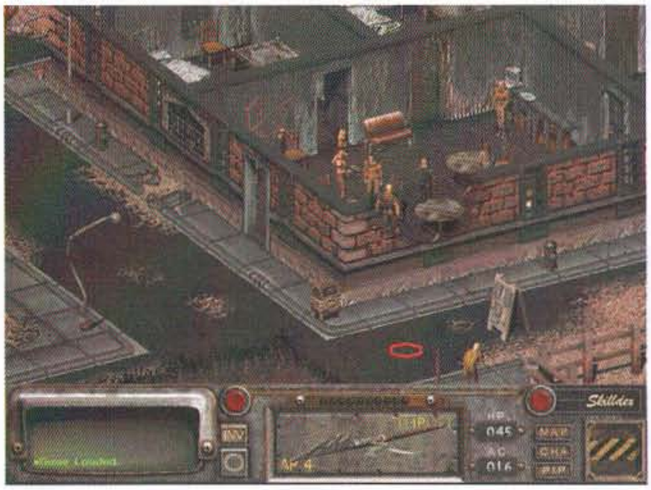
AT YOUR KEYBOARD, EVERYONE CAN HEAR YOU SCREAM



Aliens vs. Predator 2 is the stunning sequel to the original game that released on the Mac in 2001. This game takes you

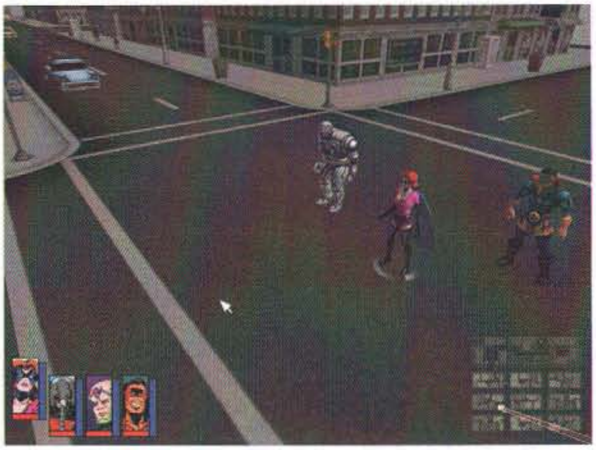
to planet LV1201 to continue the battle as either a killer Alien, the powerful Predator, or the well armed Colonial Marine. AvP2 is just as frightening and spellbinding as the first one was. Play it in a dark room with surround sound, and you'll swear you can feel the breath of the bad guys on your neck. Truly a great game. And, if you never had a chance to play the first installment, you can buy it directly form MacPlay for five bucks.

Roll playing in a new way



Many folks are addicted to these fantasy, and Dungeons and Dragons style games. The one thing so many of them have in common is the time and place they are set in. Far in the past, in the mystical land of something-or-other. Not so with the Fallout series. The first came to the Mac for OS 9, and was a big hit, in part because it was different from all the others in the genre. Fallout was set in the not too distant future, taking place in a post-apocalyptic Southern California. Fallout 2 was recently released for OS X, carrying forward the story. Beyond that, though, there is very little to differentiate part two from part one. The graphics were not improved for the sequel, and the game play is identical. If you enjoy this genre, and are not concerned with top of the line graphics, then this game is for you. Also, you can add the original Fallout to your collection, as it has been re-released for OS X and offered by MacPlay in their Value Series.

Patriot Games



Freedom Force is yet another genre of game that has been given a new, and refreshing twist. This third person perspective game, that actually incorporates both 3D role playing and strategy into the game play, takes you back to a decade filled with colorful super heroes, and fiendish villains. Set in Patriot City, you command the forces of good set against an other worldly evil bent on world domination. I know, sounds like any four b-movies you've ever seen, but this game is truly enjoyable. Another title well worth your play time.

Idle Time

All of the games we've talked about up until now have fairly big costs in time and commitment attached to them. If you don't have that kind of desire to play, don't have the time, or just don't want to burn the brain cells on high speed games, consider the following. First up is The Super GameHouse Solitaire Collection. Twenty various solitaire card games. A dream come true for card players. The games look good, and play quickly on nearly any hardware. Next is Snowball Run, an odd little game in which you steer a penguin on a snowball around various tracks. You are trying to get the little guy from one end to the other as quickly as possible, while also running over various goodies. This game really didn't do it for me. It looks nice, but just never captured my attention. Two other titles that fit in this category come on one CD. High Roller and Power Chips are both Las Vegas styled puzzle games. Extremely simple to learn, and nearly as addictive. Think about how badly Tetris sucked you in, and you've just about got it. In Power Chips, you click away at groups of the same chips (must be three or more). For High Roller, you click on two adjacent boxes to switch them to get a combination of three or more identical boxes in a row. Real easy to play, and real easy to lose time playing.

End Game

These titles are just some of the many games available from MacPlay. Nearly everyone will be able to find something that will appeal to them from the array of titles on hand. Be sure to take a look at their Value Series. MacPlay has put up some popular, older titles, for a really great price. Enjoy!

www.macplay.com

Peachpit

Essential books for the creative community

Get MX 2004 Training Straight from the Source!

Master Macromedia's new MX 2004 line-up with this best-selling series from Macromedia Press.

Macromedia Flash MX 2004: Training from the Source
By Jen deHaan
0-321-21342-4 • $44.99

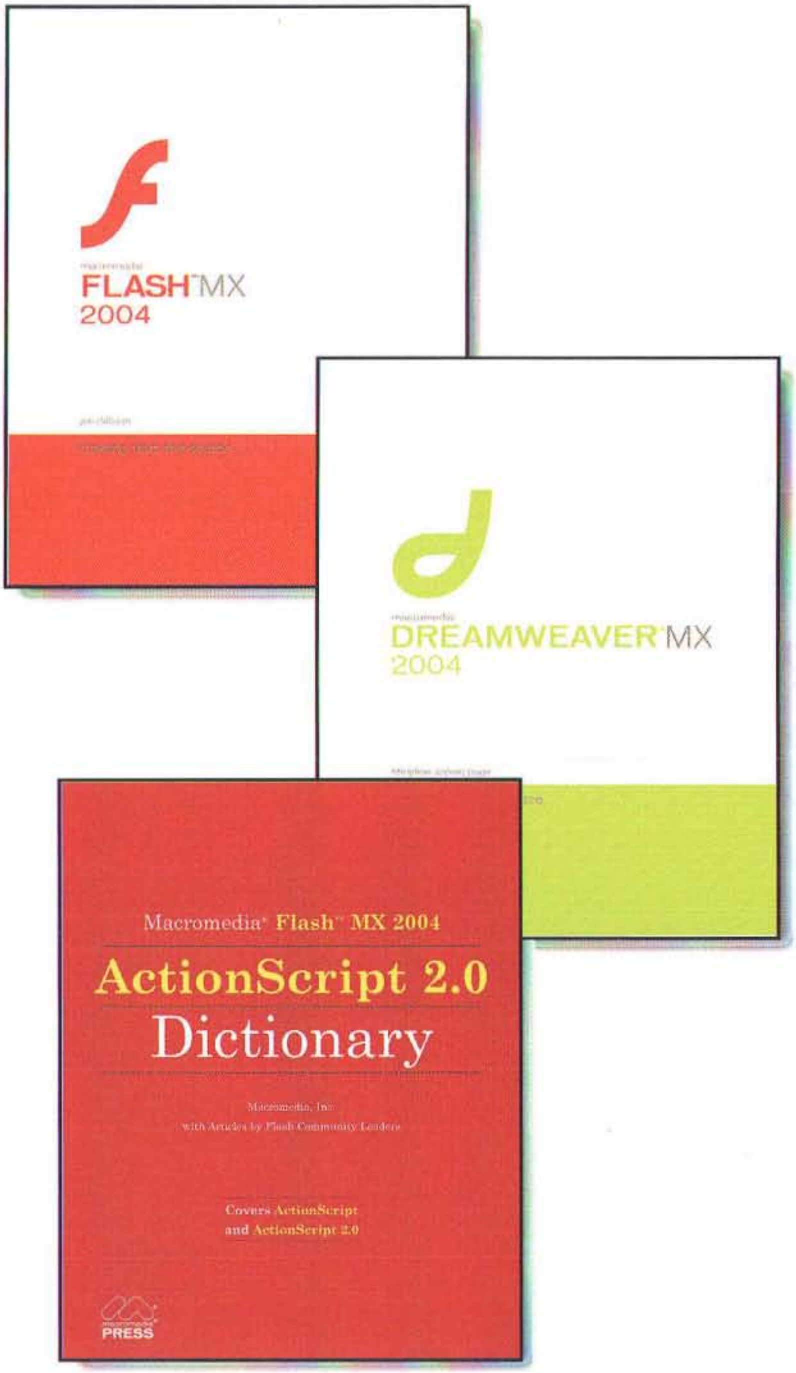
Macromedia Flash MX 2004 ActionScript: Training from the Source
By Derek Franklin and Jobe Makar
0-321-21343-2 • $44.99

Macromedia Dreamweaver MX 2004: Training from the Source
By Khristine Annwn Page
0-321-21919-8 • $44.99

Macromedia Fireworks MX 2004: Training from the Source
By Patti Schulze
0-321-21340-8 • $44.99

All New ActionScript Dictionary!

Macromedia Flash MX 2004 ActionScript 2.0 Dictionary
By Macromedia, Inc.
0-321-22841-3 • $24.99



Save up to 30%! Become a Peachpit Club Member today!

Enjoy 10% off all books every day at peachpit.com, earn an additional 10% discount as a Peachpit Club Member, and save 10% on top of that with this one-time coupon! Simply go to www.peachpit.com/mactech1003 and enter coupon code EM-O3AA-MTMF at checkout. It's that easy!

Peachpit Press

macromedia PRESS

By Michael R. Harvey

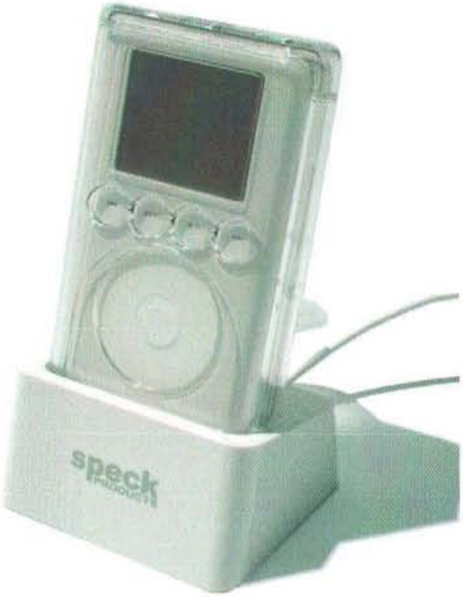
Speck Products FlipStand

Their Sophomore Effort is an iPod accessory homerun

Speck Products has a nice collection of iPod accessories available. Most are various protective, or carrying cases, for your iPod. The one I am looking at for this review is, I think, the best of their offerings, not to mention the best of all offerings for the latest generation of iPods.

Speck really knocked one out of the park with their newest FlipStand. This case is near perfect. Solid, clear plastics protect the iPod from scratch and impact damage. The buttons are easily accessible, as is the scroll wheel, for the most part (I'll get to that later). The top door easily opens to allow you to insert or remove the iPod, but still holds securely shut. You don't need to worry about it flying out of the case. There is a detachable belt clip. And, there is the flip cover that doubles as both a protector for the scroll wheel, and a stand arm to allow the case to rest at a nice viewing angle on its own.

Okay, so that's the short tour of the case, and it's very cool. It is not, however, the coolest part of this rig. The coolest bit is the dock. Dock!? Why would you need a dock when the new iPods come with one? Well, because this dock can hold the iPod while it's still in the case. This is a first. Until now, you had to remove your iPod from any covers in order to plug it into a dock. Even the latest iPod docks only have space for the iPod itself. As the FlipStand adds about 1/5 inch to the width, and 1/3 inch to the thickness of the iPod, it just won't fit in the Apple dock while in the case. The case that Speck provides with the FlipStand is specifically designed to handle iPod and case. It is weighted to keep it stable on your desk with an iPod inserted, and has a slot to hold Apple's connecting cable to allow you to plug the iPod into your computer system when to put it in the dock. This is exactly the kind of thing someone needed to bring to the iPod accessory market. The main thing that *always* drove me nuts with the previous version iPods was having to yank it out of the case to use any third party docks (like the Transpod or iPodDock). Kudos to Speck for taking care of this glaring omission.



That was the best part, so what's the worst? Remember my mention of getting to the scroll wheel accessibility issue? That is the one thing I don't like about this case. The flip cover of the case that protects the scroll wheel gets in the way of using it. The volume of individual songs in my tunes collection tends to vary (despite the equalizer in iTunes), and I found it a pain to have to flip the cover out of the way to turn the volume down when one of the blasters caught me. My ears were usually ringing by then. I found, however, that the cover was easily removed from the case, without damaging it either. With a little care, the cover can be pulled off the posts that hold it to the main body. My problem was solved. Your mileage may vary. If you find the utility of the stand to be something you can't live without, leave the case connected.

Speck Products first FlipStand was only so-so. It felt flimsy, was hard to open, and hid away the good looks of the iPod. This new FlipStand reminds me of Contour Designs case for the last generation iPods, the iSee. Clear plastics that protect the iPod while simultaneously showing off its great style. The iSee was the best of the last generation cases, and now the FlipStand has taken that mantle and run with it. Good looks, high durability, and the dock make this accessory the one to get for your third generation iPod. And that stuff about the flip cover, well, it's subjective, and easily worked around. This product is well worth the $29.95 price tag Speck is offering them at on their web site. Get one of these. You'll be happy you did.

www.speckproducts.com

Time Track

Keep track of every second of your time and bill for it with Time Track! Built in instructions make it easy to use. It is a simple way to manage your billable time for multiple projects and create a web page to show to your clients. Only $24.95 per single user license per platform. Finally! A versatile time tracking solution for Macintosh, Windows, and Palm!

www.trinfinitysoftware.com

Trapcode - plug-ins for Adobe® After Effects®

Trapcode Shine is a fast light effect plug-in. The effect looks very much like volumetric light, but is actually a 2D effect. There are special controls to make shimmering lights and numerous coloring modes. This is an effect that you see everyday on TV and in many movie titles. Shine is available for Mac, Mac OSX and Windows.

www.trapcode.com

dragTHING

By TLA Systems

The Dock with more than one dimension.

Create multiple docks of any size, assign hot keys and even put the Trash back on the Desktop. A flexible and feature-laden tool for power-users. Runs natively on Mac OS X and 9 in five languages.

"...you made the switch to OS X a lot easier for me..." - Bob LeVitus

"...DragThing can rightfully be called an indispensable aid to working with your Mac..." - MacUser UK

Download a copy now from www.dragthing.com.

EUDORA INTERNET MAIL SERVER

Eudora Internet Mail Server (EIMS) 3.2 is the latest version of the most popular Internet mail server for the Macintosh. If you need to handle email for a dozen users, or thousands of users, EIMS is a reliable and easy to use solution.

EIMS 3.2 is available for US$400.00, there are no limits on the number of users that can be added, and free email support is included.

For more information, see
http://www.eudora.co.nz/

GraphicConverter converts pictures to different formats. Also it contains many useful features for picture manipulation.

See ***www.lemkesoft.com*** for more information.

ADVERTISER/ PRODUCT LIST

List of Advertisers

Aladdin Systems, Inc.	11
Big Nerd Ranch, Inc.	57
Brad Sniderman	43
Circus Ponies Software, Inc.	45
DevDepot	28-29
EazyDraw (Dekorra Optics, LLC)	53
Electric Butterfly	38
Eudora Internet Mail Server	79
FairCom Corporation	1
Felt Tip Software	10
Fetch Softworks	37
Full Spectrum Software, Inc.	19
IDG World Expo Corporation	50-51
James Sentman Software	56
Lemke Software GmbH	79
Lingo Systems	55
MacDirectory	25
MacTech Magazine	71
Mathemaesthetics, Inc.	15
MCF Software	27
MYOB US, Inc.	49
Netopia, Inc.	IFC
Paradigma Software	31
Peachpit Press	77
Pearson Education Communications	59
piDog Software	65
PowerGlot Software	61
Presto Vivace, Inc.	23
PrimeBase (SNAP Innovation)	47
Prosoft Engineering, Inc.	63
RadGad	75
Redstone Software, Inc.	39
Runtime Revolution Limited	BC
Seapine Software, Inc.	17
Small Dog Electronics	69
Sophisticated Circuits, Inc.	33
Sophos, Inc.	9
Sybase, Inc.	2-3
TechSmith Corporation	7
The Iconfactory	43
ThinkFree Corporation	21
Trango Broadband Wireless	35
Trolltech AS	13
Utilities4Less.com	73
VVI	41
WIBU-SYSTEMS AG	IBC

List of Products

Adobe Press • Peachpit Press	77
Big Nerd Ranch • Big Nerd Ranch, Inc.	57
c-tree Plus • FairCom Corporation	1
Developers Library • Pearson Education Communications	59
Development & Testing • Full Spectrum Software, Inc.	19
Disk Utilities • Prosoft Engineering, Inc.	63
Eazy Draw • EazyDraw (Dekorra Optics, LLC)	53
Eggplant • Redstone Software, Inc.	39
EIMS • Eudora Internet Mail Server	79
Ensharpen • TechSmith Corporation	7
Enterprise Software • MCF Software	27
Fetch • Fetch Softworks	37
Graphic Converter • Lemke Software GmbH	79
HelpLogic • Electric Butterfly	38
InstallerMaker, StuffIt • Aladdin Systems, Inc.	11
Law Offices • Brad Sniderman	43
Long Distance Phone Service • Utilities4Less.com	73
MacDirectory • MacDirectory	25
MacTech Magazine Subscription • MacTech Magazine	71
Macworld Conference & Expo • IDG World Expo Corporation	50-51
Maximizing Your Mac! • DevDepot	28-29
New Product • MYOB US, Inc.	49
Notebook • Circus Ponies Software, Inc.	45
piDog Utilities • piDog Software	65
PowerGlot • PowerGlot Software	61
PowerKey Pro & KickOff! • Sophisticated Circuits, Inc.	33
Presto Vivace Services • Presto Vivace, Inc.	23
PrimeBase • PrimeBase (SNAP Innovation)	47
Qt • Trolltech AS	13
Resorcerer • Mathemaesthetics, Inc.	15
Runtime Revolution • Runtime Revolution Limited	BC
SmallDog.com • Small Dog Electronics	69
Software Protection • WIBU-SYSTEMS AG	IBC
Sophos Anti-Virus • Sophos, Inc.	9
Sound Studio • Felt Tip Software	10
Stock Icons • The Iconfactory	43
Sybase • Sybase, Inc.	2-3
TestTrack Pro • Seapine Software, Inc.	17
ThinkFree • ThinkFree Corporation	21
Timbuktu & netOctopus • Netopia, Inc.	IFC
Translation & Localization • Lingo Systems	55
Useful Gifts & Gadgets • RadGad	75
Valentina • Paradigma Software	31
Visual-Report Tool Developer • VVI	41
WhistleBlower • James Sentman Software	56
Wireless Networking • Trango Broadband Wireless	35

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

WIBU-KEY Software Protection

Be sure to order a **WIBU-KEY Protection Kit** and find out why small software companies to Fortune 500 enterprises are switching to the WIBU-KEY **Security, not Obscurity** model for all of their software licensing needs, including:

- **Common API across all platforms and hardware**
- **Hardware-based code and data encryption**
- **Field-upgradable and reusable hardware**
- **The most cost-effective network licensing solution available**
- **The only system to employ RID/RED and AXAN security**
- **Engineered and manufactured to exacting ISO9001 standards**

Test the
WIBU-KEY
Protection Kit
sales@griftech.com

Call 800-986-6578

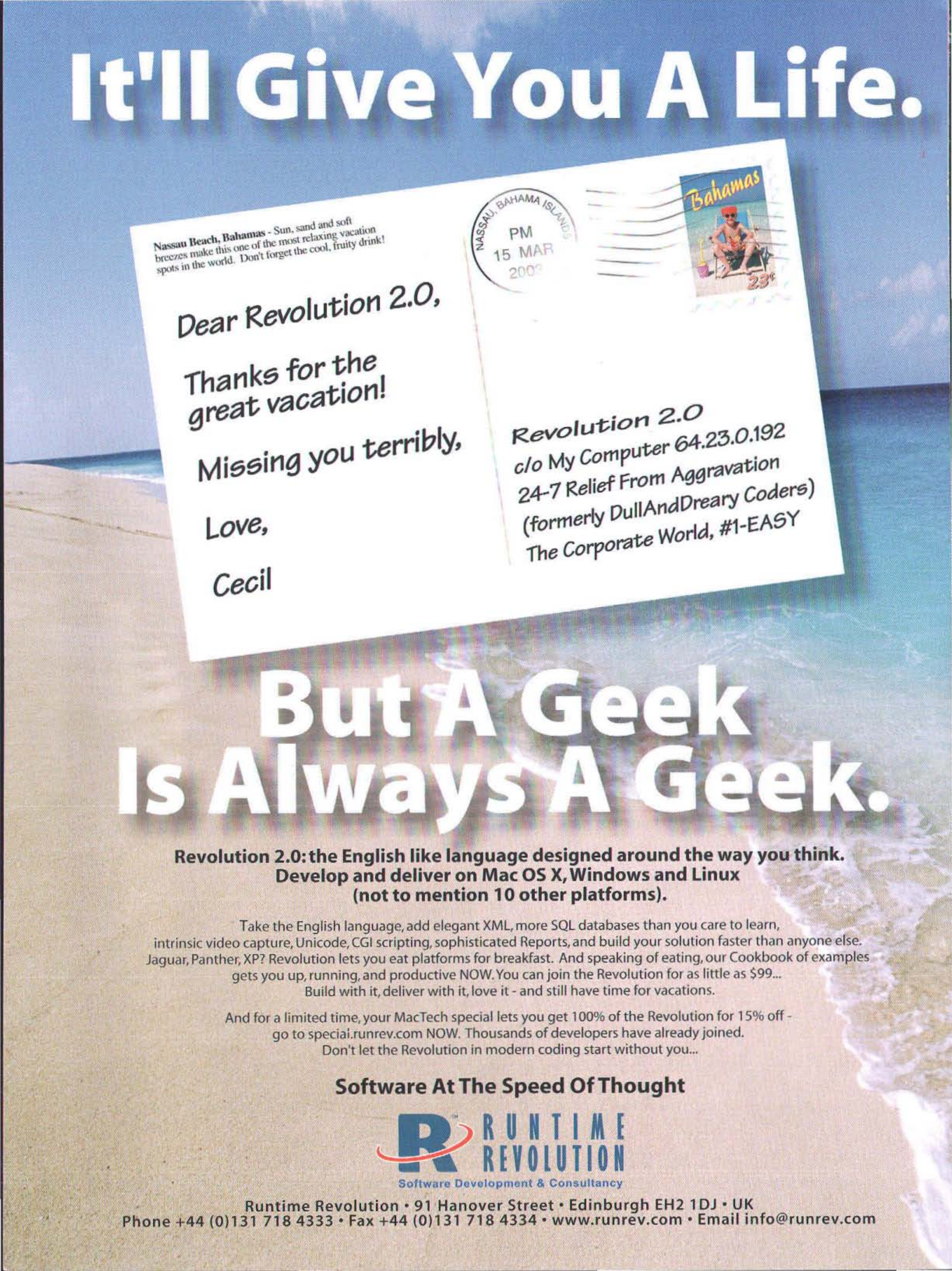
The Key is in Your Hands!

WIBU
SYSTEMS

WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
Email: info@wibu.com

www.griftech.com
www.wibu.com

Protection Kits also available at:
Belgium wibu@impakt.be, **Denmark** lean@danbit.dk, **Finland** finebyte@finebyte.com, **France** info@neol.fr, **Hungary** info@mrsoft.hu, **Japan** info@suncarla.co.jp, **Jordan, Lebanon** starsoft@cyberia.net.lb, **Korea** dhkimm@wibu.co.kr, **Luxembourg** wibu@impakt.be, **Netherlands** wibu@impakt.be, **Portugal** dubit@dubit.pt, **Thailand** preecha@dptf.co.th, **United Kingdom** info@codework.com, **USA** sales@griftech.com



It'll Give You A Life.

Nassau Beach, Bahamas - Sun, sand and soft breezes make this one of the most relaxing vacation spots in the world. Don't forget the cool, fruity drink!

Dear Revolution 2.0,

Thanks for the great vacation!

Missing you terribly,

Love,

Cecil

Revolution 2.0
c/o My Computer 64.23.0.192
24-7 Relief From Aggravation
(formerly DullAndDreary Coders)
The Corporate World, #1-EASY

But A Geek Is Always A Geek.

**Revolution 2.0: the English like language designed around the way you think.
Develop and deliver on Mac OS X, Windows and Linux
(not to mention 10 other platforms).**

Take the English language, add elegant XML, more SQL databases than you care to learn, intrinsic video capture, Unicode, CGI scripting, sophisticated Reports, and build your solution faster than anyone else. Jaguar, Panther, XP? Revolution lets you eat platforms for breakfast. And speaking of eating, our Cookbook of examples gets you up, running, and productive NOW. You can join the Revolution for as little as $99... Build with it, deliver with it, love it - and still have time for vacations.

And for a limited time, your MacTech special lets you get 100% of the Revolution for 15% off - go to special.runrev.com NOW. Thousands of developers have already joined. Don't let the Revolution in modern coding start without you...

Software At The Speed Of Thought

RUNTIME REVOLUTION
Software Development & Consultancy

Runtime Revolution • 91 Hanover Street • Edinburgh EH2 1DJ • UK
Phone +44 (0)131 718 4333 • Fax +44 (0)131 718 4334 • www.runrev.com • Email info@runrev.com